

# **Introduction to Engineering & Computer Science**

*“with emphasis on Computing and Electrical Fields”*

*Version 6.0 printed on January 2021  
First published in September 2006*

## **Background and Acknowledgements**

This text is prepared for students who intend to start their education in Engineering or Computer Science fields with emphasis on Electrical Engineering, Computer Engineering and Computer Science. The content is derived from the author's educational, technical and management experiences, in-addition to teaching experience and colleagues. Many other sources, including the following specific sources, have also informed the content and format of this text:

- Katz, R. Contemporary Logic Design. (2005) Pearson.
- Lumsdaine, E. Creative Problem Solving and Engineering. (1999) Prentice Hall.
- Wakerley, I. Digital Design. (2001) Prentice Hall.
- Sandige, R. Digital Design Essentials. (2002) Prentice Hall.
- Nilsson, J. Electrical Circuits. (2004) Pearson.
- Eide, A. Engineering Fundamentals & Problem Solving. (2002) McGraw Hill
- MathWorks. MATLAB Reference Material Version R2000a. (2007) MathWorks

I would like to give thanks to my students and colleagues for their valued contributions in making this material a more effective learning tool. Special thanks to Jim Teisher, Jing Liu, Opinder Behlla and Malcolm Anderson for their contributions.

I invite the reader to forward any corrections, additional topics, examples and problems to me for future updates.

Thanks,

*Izad Khormae*

[www.EngrCS.com](http://www.EngrCS.com)

*Copyright 2020 Izad Khormae, All Rights Reserved.*

## Table of Contents

<b>Chapter 1. Introduction.....</b>	<b>5</b>
1.1 Key Concepts and Overview.....	5
1.2. Engineering and Computer Science (ECS) .....	6
1.3. Education .....	7
1.4. Careers .....	9
1.5. Achievements and Benefits .....	11
1.6. Societies and Certification.....	13
1.7. Key Success Factors .....	14
1.8. Patents .....	15
1.9. Code of Ethics .....	16
1.10. Additional Resources .....	21
1.11. Problems .....	22
<b>Chapter 2. Teamwork and Communication.....</b>	<b>24</b>
2.1. Key Concepts and Overview.....	24
2.2. Thinking Modes.....	25
2.3. Thinking Mode Assessment Based on Herrmann Brain Dominance Model (HBDM).....	28
2.4. Teamwork .....	30
2.5. Communication .....	33
2.6. Additional Resources .....	37
2.7. Problems .....	38
<b>Chapter 3. Creative Problem Solving .....</b>	<b>39</b>
3.1. Key Concepts and Overview.....	39
3.2. Creative Problem-Solving Process .....	40
3.3. Step 1 - Customer Issues/Needs Identification.....	42
3.4. Step 2 - Problem Definition .....	43
3.5. Step 3 - Idea Generation.....	45
3.6. Step 4 - Optimal Solution Selection .....	48
3.7. Step 5 - Solution Implementation.....	52
3.8. Additional Resources .....	53
3.9. Problems .....	54
<b>Chapter 4. Electrical Circuits .....</b>	<b>55</b>
4.1. Key Concepts and Overview.....	55
4.2. Charge, Current and Voltage .....	56
4.3. Ideal Circuit Model .....	59
4.4. Power Calculation .....	60
4.5. Resistor Simplifications.....	66
4.6. Common Terms Used in Circuit Analysis & Examples.....	75
4.7. Kirchhoff's Current Laws (KCL) .....	79
4.8. Additional Resources .....	89
4.9. Problems .....	90
<b>Chapter 5. Digital Logic .....</b>	<b>94</b>
5.1. Key Concepts and Overview.....	94
5.2. Digital vs. Analog .....	95
5.3. Digital Design Overview.....	97
5.4. Binary Number Systems .....	98
5.5. Standard Logic Gates & Binary Algebra .....	101
5.6. Input and Output Configurations .....	105

5.7. Introduction to Logic Circuit Design .....	109
5.8. Additional Resources .....	116
5.9. Problems .....	117
<b>Chapter 6. Computer Architecture and Programming Fundamentals .....</b>	<b>119</b>
6.1. Key Concepts and Overview .....	119
6.2. Computer Architecture .....	120
6.3. Programming Levels .....	123
6.4. Common Programming Languages .....	124
6.5. Software Development Steps .....	125
<b>Chapter 7. Programming in Python .....</b>	<b>127</b>
7.1. Key Concepts and Overview .....	127
7.2. Getting Started with Python .....	128
7.3. Python Variables and Operators .....	130
7.4. Creating Python Program Files (Script) .....	135
7.5. Python Flow Control .....	137
7.6. Python Built-In Functions .....	140
7.7. Python Modules .....	143
7.8. Python User Defined Functions .....	146
7.9. Python Quick Reference .....	148
7.10. Further Reading .....	151
7.11. Problems .....	152
<b>Chapter 8. Programming in MATLAB .....</b>	<b>154</b>
8.1. Key Concepts and Overview .....	154
8.2. Development Environment Interface and Structure .....	155
8.3. Using the MATLAB Command Window .....	159
8.4. Creating and Editing M-files .....	161
8.5. MATLAB Arithmetic and Logic Operators .....	163
8.6. MATLAB Data Flow Controls .....	166
8.7. Problems .....	175
<b>Chapter 9. Mathematical Concepts .....</b>	<b>177</b>
9.1. Key Concepts and Overview .....	177
9.2. Matrices .....	178
9.3. MATLAB Matrix Operations .....	181
9.4. Trigonometry .....	184
9.5. MATLAB Trigonometry Operations .....	190
9.6. Complex Numbers .....	195
9.7. MATLAB Complex Number Operations .....	197
9.8. Additional Resources .....	198
9.9. Problems .....	199
<b>Appendix A. Open Source Alternatives to MATLAB .....</b>	<b>201</b>
<b>Appendix B. Additional Resources .....</b>	<b>202</b>

## **Chapter 1. Introduction**

### **1.1 Key Concepts and Overview**

- Engineering and Computer Science Professions
- Education
- Careers
- Achievements and Benefits
- Societies and Certifications
- Key Success Factors
- Patents
- The Code of Ethics

## 1.2. Engineering and Computer Science (ECS)

ECS professions are diverse and would require a multi-volume book to fully cover them. Engineers apply principles of science and mathematics in order to develop innovative solutions to problems. Majority of tools and systems in today's world rely on electrical/computer components that require the skills of Electrical Engineers, Computer Scientists and Computer Engineers. The need for these professionals cuts across all industries without exception. For example, an appliance, a car or an airplane have numerous computer and electrical systems that need to be designed and programmed.

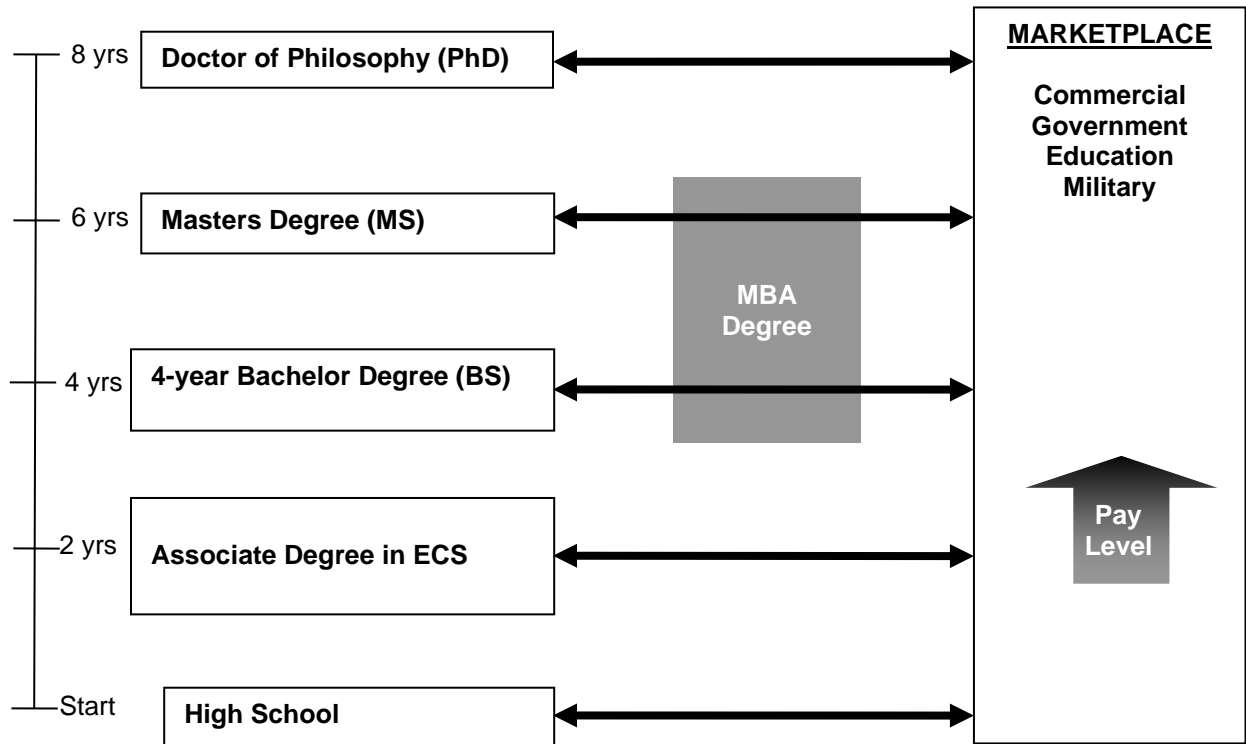
### **Student Exercise** – Benefits

Can you think of a Company that is not benefiting from Engineering and Computer Science professionals?

### **Solution**

### 1.3. Education

The Engineering and Computer Science education can be best described in a number of two-year intervals after high school completion. The following diagram shows the degree attained after approximately each two-year study and entry into the marketplace:



The program at a community college such as Clark College covers the first two years of post-high school studies resulting in an Associate Degree. At Clark College, the Associate Degree designed for Computer and Electrical Engineering is referred to as Computer and Electrical Engineering Major Ready Pathway (CEE MRP - AST2) and for Computer Science it is referred to as AST2-Computer Science. Below are the typical course requirements for the first two years degrees:

- 1) Communication, Social Science and Humanities (15 Credits)
- 2) Mathematics Courses
  - College Algebra (MATH 111), College Trigonometry (MATH 103)
  - Calculus I – IV (MATH& 151, 152, 153, 254)
  - Differential Equations (MATH 221) and Linear Algebra (MATH 215)
- 3) Engineering Physics I, II and III (Phys 2xx)
- 4) Introduction to ECSE (ENGR/CSE 120)
- 5) Electrical Circuits and Signal Processing I, II, III –(ENGR &204, 252, 253 - Electrical and Computer Engineering Only)

- 6) Digital Logic Design (ENGR 250)
- 7) Microprocessor Design (ENGR 270)
- 8) Introduction to Programming (CSE 121)
- 9) Discrete Structure (CSE 215 - Computer Science only)
- 10) Data Structure I, II (CSE 222, 223 - Computer Science Only)
- 11) Programming Tools (CSE 224 - Computer Science Only)

It is important that students work with their academic advisors to develop an education plan that is designed for their specific objectives and background as soon as possible.

### **Benefits of attending Community College**

After completing high school, students have the option of attending university or community college. The following list outlines some of the key benefits of community college studies:

- **Proven success** in the ECS Transfer Program
- **Smaller class size** and easy access to professors
- **Lower cost** – typically one third of the cost of university programs
- **Local, online and in-person classes** - no need to relocate
- Extensive **hands-on project** opportunities

### **Common Destination Universities**

After completing the first two years of studies in community college, students can transfer as juniors into a four year university to complete their Bachelor's Degree. The top transfer universities are listed in order of the number of student transfers from Clark College:

- Washington State University – Vancouver Campus
- Portland State University
- University of Washington – Seattle, Tacoma & Bothell Campuses
- University of Portland
- Oregon State University
- Washington State University – Pullman Campus



## 1.4. Careers

Careers in ECS are typically defined based on technology, location in the product development life cycle, or a mix of the two attributes. The following list outlines some of the technologies, relationships and product development life cycles for these three closely related fields:

### Technology Fields

Here are the major technology categories for each of the three fields often used to define ECS positions:

#### Electrical Engineering

- Power and Energy Systems
- Communication Systems
- Magnetic Fields
- Semiconductors
- Electronics
- Signal Processing
- Biomedical
- Control Systems
- Automation
- Sensors
- ...

#### Computer Engineering

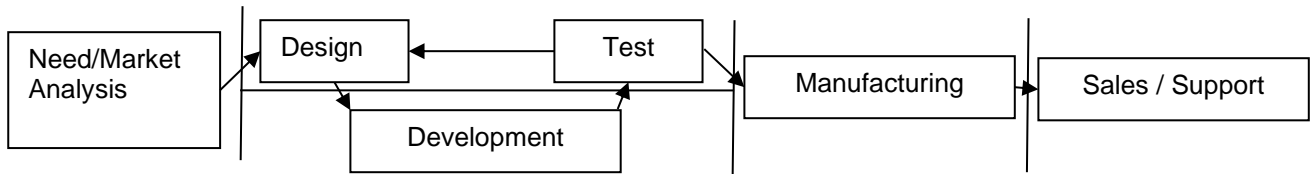
- Computer Systems
- Networking
- Embedded Systems
- Automation/Robotics
- Artificial Intelligence
- Biomedical
- ...

#### Computer Science

- Application
- Networking
- Embedded System
- Artificial Intelligence
- Game Development
- Automation/Robotics
- Computing infrastructure
- ...

### Life Cycles

Product and Service development life cycle is another way for some organizations to define ECS positions. In other words, positions are defined based on their roles within the product development life cycles. A typical product development life cycle can be outlined as:



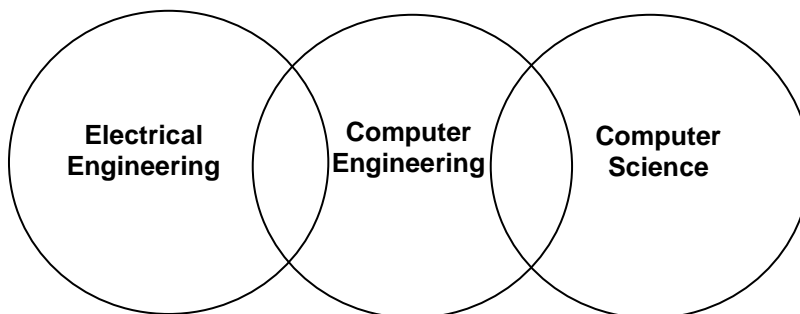
ECS positions, with focus on specific parts of the life cycle, are titled as:

- Design Engineer
- Development Engineer
- Test Engineer
- Process Engineer
- Application Engineer
- Sales Engineer
- Support Engineer
- Field or Customer Engineer

### Relationships Among the ECS Fields

There are overlaps among Electrical Engineering, Computer Engineering and Computer Science. It is common that a professional in one field to engage in cross-discipline projects. For example, in a robotic project, all three disciplines will be involved. Electrical engineers design the electrical circuit (motors, sensors, etc.) for the robot, computer scientists write the software, and computer engineers develop software and circuits required to unify the electrical and software portions of the project.

The following Venn Diagram depicts the overlap and relationships among the three fields:



## 1.5. Achievements and Benefits

This section reflects on past achievements and allows students to imagine the possibilities of the next 10 years. Finally, careers are selected based on benefits of the career and one's interest which are discussed at the end of this Section.

### **Greatest Engineering and Computer Science (ECS) Achievements in the Past 100 Years**

- #10 - Space Explorations
- # 9 - High Performance Material
- # 8 - Genetic and Bio Engineering
- # 7 - Health and Medical Technology
- # 6 - Transportation (Automobile, Airplane)
- # 5 - Communication Technology (Phone, Radio/Television, Internet)
- # 4 - Electronics and Computers
- # 3 - Air Conditioning and Refrigeration
- # 2 - Agricultural Mechanization
- # 1 - Easy Access to Safe and Drinkable Water

### **Student Exercise – Achievements**

List the top ten ECS achievements you expect will come about in the next decade

### **Solution**

- #10 -
- # 9 -
- # 8 -
- # 7 -
- # 6 -
- # 5 -
- # 4 -
- # 3 -
- # 2 -
- # 1 -

### **Benefits of an Engineering Career:**

- **Broad Range of Opportunities and Growth Potential** – ECS education is a great foundation for a broad range of technical and management positions. ECS professionals are problem-solvers and although most start solving technical problems, over time, their careers may take in many different directions. ECS educated professionals are found working as Designers, Developers, Managers and even Chief Executive Officers (CEO).
- **Intellectually Challenging Work** – ECS professionals are employed to solve problems that have not been previously solved. This means the projects are new or a variation of a past problem but always interesting. Additionally, opportunities are limitless and only gated by the willingness and ability of the professional.
- **Adding Value to Society and Being Financially Rewarded** - Engineers comprise about 1% of population but directly affect over 90% of society's production capacity. It is impossible to think of a product or service that has not benefited from the contributions of Engineers and Computer Scientists.

An ECS career is financially rewarding, with a starting salary typically over \$60,000 per year. Additionally, the work environment is safe, comfortable and staff supported.

### **Student Exercise – Expectations**

What are the top three benefits that you desire from your career? Does an ECS career provide you with these desired benefits?

**Solution**

## 1.6. Societies and Certification

This Section covers the need to get involved in Professional Societies/Clubs and introduces the Professional Engineer (PE) certification.

### Professional Societies/Clubs

Societies and clubs help ECS students share ideas, get hands-on experience, and meet professionals. Additionally, the experience gained in leadership and technical projects plays an important role in finding and getting your first internship or job.

Students are encouraged to consider joining and becoming active in one or more of the following groups:

- Clark College Engineering and Computer Science (ECS) Club
- Institute of Electrical and Electronics Engineering (IEEE)
- Association for Computing Machinery (ACM)

### Professional Engineer (PE) Certification

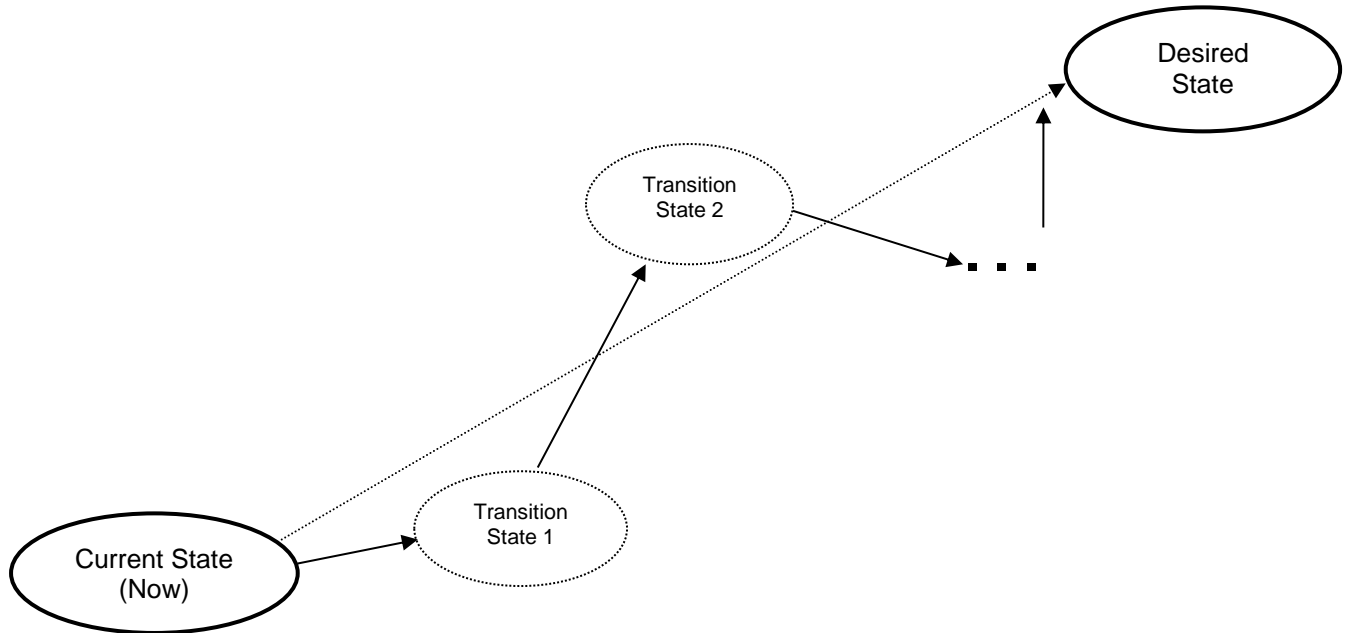
Engineers may obtain a Professional Engineer (PE) Certification by completing the following requirements:

- Completion of an engineering degree from an ABET-credited College
- Passing the “Engineer In Training” (EIT) Exam
- Completing four years of practice under the supervision of a professional engineer
- Passing the “Principles and Practice of Engineering” PE examination

PE Certification is important for Engineers who will be working in areas directly effecting public safety such as Civil and Structural Engineers who are in private practice.

## 1.7. Key Success Factors

Success is not an accident; it is the result of careful planning and hard work. Successful people know that success is driven by planning, persistence and focus on goals (Desired State). Below is a formalized model for setting up long-term goals and the intermediate steps needed in order to achieve goals:



The above model is referred to as the Current-Desired State Model or Diagram. This model helps focus the person or the organization on long-term goals and ensures that all activities are evaluated based on contributions to the long-term goal (Desired State) in addition to immediate values.

Once the long-term plan is developed, the next step is to consider the factors for successful execution. The most important factor is a positive attitude, which simply means seeing success as reaffirmation of the direction, and failure as the opportunity to learn, adjust the plan and try again.

The next item to consider is “working smart”, which means when investing resources on a task, consider the impact on your success. If the task is not producing the desired result, then adjust your approach to improve effectiveness. We must make the most of our limited time and resources.

## 1.8. Patents

Typically, engineers are either given or find problems that, if solved, add value to society. The Patent Office was created to protect the right of inventors to exclusively benefit from their inventions.

Any idea, process, procedure, system, or design may be patented as long as it passes the following two tests:

- 1) Novel  
The “novel” test is passed if the topic being patented is not described in an existing document or patent.
- 2) Non-Obvious  
The “non-obvious” test is passed if the topic being patented cannot be described by multiple existing documents or patents.

The Patent Office is given the responsibility to evaluate and approve patents. When applying for a patent, it is important to answer the following questions:

- What is the problem and who will benefit from the proposed solution?
- What are the new inventions being claimed by this patent? (Only new ideas in the solutions being claimed are referred to as the claims)
- What are the existing patents that relate to the claim(s) being made?

The law allows for protection of other types of intellectual property such as:

- Copyright ©  
Any written material can be protected through Copyright process.
- Trademark  
Images associated with a product can be protected through Trademark process.
- Service mark  
Images associated with a service can be protected through Service mark process.

More information on the patent application is available at the Patent Office online website at [www.uspto.gov](http://www.uspto.gov).

## 1.9. Code of Ethics

Each profession is guided by a code of ethics which is intended to help members conduct their professional duties in an ethical way. Ethical performance benefits the profession and its members in the long run.

### Definition of Ethics according to dictionary.com

Ethics (*used with a sing. or pl. verb*) the rules or standards governing the conduct of a person or the members of a profession:

- A set of principles of right conduct.
- A theory or a system of moral values: "An ethic of service is at war with a craving for gain" (Gregg Easterbrook).

### Stakeholders

Stakeholders are individuals and groups who are in the best position to impact the success of program and also, they are the ones who the program most directly impacts. In a typical program or project, the following groups are typically included as stakeholders with respect to ethics:

- Self
- Employer
- Public
- Client
- Users

Here, the National Society of Professional Engineers (NSPE) Code of Ethics for Engineers (Revised January 2003/ Reprint permission granted on 1/2004) is used as an example of an engineering code of ethics.

### Preamble

Engineering is an important and learned profession. As members of this profession, engineers are expected to exhibit the highest standards of honesty and integrity. Engineering has a direct and vital impact on the quality of life for all people. Accordingly, the services provided by engineers require honesty, impartiality, fairness, and equity, and must be dedicated to the protection of the public health, safety, and welfare. Engineers must perform under a standard of professional behavior that requires adherence to the highest principles of ethical conduct.

### Fundamental Canons

Engineers in the fulfillment of their professional duties shall:

- 1) Hold paramount the safety, health and welfare of the public.
- 2) Perform services only in areas of their competence.
- 3) Issue public statements only in an objective and truthful manner.
- 4) Act for each employer or client as faithful agents or trustees.
- 5) Avoid deceptive acts.
- 6) Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.



## Rules of Practice

- 1) Engineers shall hold paramount the safety, health, and welfare of the public.
  - a) If engineers' judgment is overruled under circumstances that endanger life or property, they shall notify their employer or client and such other authority as may be appropriate.
  - b) Engineers shall approve only those engineering documents that are in conformity with applicable standards.
  - c) Engineers shall not reveal facts, data, or information without the prior consent of the client or employer except as authorized or required by law or this Code.
  - d) Engineers shall not permit the use of their name or associate in business ventures with any person or firm that they believe are engaged in fraudulent or dishonest enterprise.
  - e) Engineers shall not aid or abet the unlawful practice of engineering by a person or firm.
  - f) Engineers having knowledge of any alleged violation of this Code shall report thereon to appropriate professional bodies and, when relevant, also to public authorities, and cooperate with the proper authorities in furnishing such information or assistance as may be required.
- 2) Engineers shall perform services only in the areas of their competence.
  - a) Engineers shall undertake assignments only when qualified by education or experience in the specific technical fields involved.
  - b) Engineers shall not affix their signatures to any plans or documents dealing with subject matter in which they lack competence, nor to any plan or document not prepared under their direction and control.
  - c) Engineers may accept assignments and assume responsibility for coordination of an entire project and sign and seal the engineering documents for the entire project, provided that each technical segment is signed and sealed only by the qualified engineers who prepared the segment.
- 3) Engineers shall issue public statements only in an objective and truthful manner.
  - a) Engineers shall be objective and truthful in professional reports, statements, or testimony. They shall include all relevant and pertinent information in such reports, statements, or testimony, which should bear the date indicating when it was current.
  - b) Engineers may express publicly technical opinions that are founded upon knowledge of the facts and competence in the subject matter.
  - c) Engineers shall issue no statements, criticisms, or arguments on technical matters that are inspired or paid for by interested parties, unless they have prefaced their comments by explicitly identifying the interested parties on whose behalf they are speaking and by revealing the existence of any interest the engineers may have in the matters.
- 4) Engineers shall act for each employer or client as faithful agents or trustees.
  - a) Engineers shall disclose all known or potential conflicts of interest that could influence or appear to influence their judgment or the quality of their services.
  - b) Engineers shall not accept compensation, financial or otherwise, from more than one party for services on the same project, or for services pertaining to the same project, unless the circumstances are fully disclosed and agreed to by all interested parties.
  - c) Engineers shall not solicit or accept financial or other valuable consideration, directly or indirectly, from outside agents in connection with the work for which they are responsible.
  - d) Engineers in public service as members, advisors, or employees of a governmental or quasigovernmental body or department shall not participate in decisions with respect to services solicited or provided by them or their organizations in private or public engineering practice.
  - e) Engineers shall not solicit or accept a contract from a governmental body on which a principal or officer of their organization serves as a member.

- 5) Engineers shall avoid deceptive acts.
  - a) Engineers shall not falsify their qualifications or permit misrepresentation of their or their associates' qualifications. They shall not misrepresent or exaggerate their responsibility in or for the subject matter of prior assignments. Brochures or other presentations incident to the solicitation of employment shall not misrepresent pertinent facts concerning employers, employees, associates, joint ventures, or past accomplishments.
  - b) Engineers shall not offer, give, solicit, or receive, either directly or indirectly, any contribution to influence the award of a contract by public authority, or which may be reasonably construed by the public as having the effect of intent to influencing the awarding of a contract. They shall not offer any gift or other valuable consideration to secure work. They shall not pay a commission, percentage, or brokerage fee to secure work, except to a bona fide employee or bona fide established commercial or marketing agencies retained by them.

### **Professional Obligations**

- 1) Engineers shall be guided in all their relations by the highest standards of honesty and integrity.
  - a) Engineers shall acknowledge their errors and shall not distort or alter the facts.
  - b) Engineers shall advise their clients or employers when they believe a project will not be successful.
  - c) Engineers shall not accept outside employment to the detriment of their regular work or interest. Before accepting any outside engineering employment, they will notify their employers.
  - d) Engineers shall not attempt to attract an engineer from another employer by false or misleading pretenses.
  - e) Engineers shall not promote their own interest at the expense of the dignity and integrity of the profession.
- 2) Engineers shall at all times strive to serve the public interest.
  - a) Engineers shall seek opportunities to participate in civic affairs; career guidance for youths; and work for the advancement of the safety, health, and well-being of their community.
  - b) Engineers shall not complete, sign, or seal plans and/or specifications that are not in conformity with applicable engineering standards. If the client or employer insists on such unprofessional conduct, they shall notify the proper authorities and withdraw from further service on the project.
  - c) Engineers shall endeavor to extend public knowledge and appreciation of engineering and its achievements.
- 3) Engineers shall avoid all conduct or practice that deceives the public.
  - a) Engineers shall avoid the use of statements containing a material misrepresentation of fact or omitting a material fact.
  - b) Consistent with the foregoing, engineers may advertise for recruitment of personnel.
  - c) Consistent with the foregoing, engineers may prepare articles for the lay or technical press, but such articles shall not imply credit to the author for work performed by others.
- 4) Engineers shall not disclose, without consent, confidential information concerning the business affairs or technical processes of any present or former client or employer, or public body on which they serve.
  - a) Engineers shall not, without the consent of all interested parties, promote or arrange for new employment or practice in connection with a specific project for which the engineer has gained particular and specialized knowledge.

- b) Engineers shall not, without the consent of all interested parties, participate in or represent an adversary interest in connection with a specific project or proceeding in which the engineer has gained particular specialized knowledge on behalf of a former client or employer.
- 5) Engineers shall not be influenced in their professional duties by conflicting interests.
- a) Engineers shall not accept financial or other considerations, including free engineering designs, from material or equipment suppliers for specifying their product.
  - b) Engineers shall not accept commissions or allowances, directly or indirectly, from contractors or other parties dealing with clients or employers of the engineer in connection with work for which the engineer is responsible.
- 6) Engineers shall not attempt to obtain employment or advancement or professional engagements by untruthfully criticizing other engineers, or by other improper or questionable methods.
- a) Engineers shall not request, propose, or accept a commission on a contingent basis under circumstances in which their judgment may be compromised.
  - b) Engineers in salaried positions shall accept part-time engineering work only to the extent consistent with policies of the employer and in accordance with ethical considerations.
  - c) Engineers shall not, without consent, use equipment, supplies, laboratory, or office facilities of an employer to carry on outside private practice.
- 7) Engineers shall not attempt to injure, maliciously or falsely, directly or indirectly, the professional reputation, prospects, practice, or employment of other engineers. Engineers who believe others are guilty of unethical or illegal practice shall present such information to the proper authority for action.
- a) Engineers in private practice shall not review the work of another engineer for the same client, except with the knowledge of such engineer, or unless the connection of such engineer with the work has been terminated.
  - b) Engineers in governmental, industrial, or educational employ are entitled to review and evaluate the work of other engineers when so required by their employment duties.
  - c) Engineers in sales or industrial employ are entitled to make engineering comparisons of represented products with products of other suppliers.
- 8) Engineers shall accept personal responsibility for their professional activities, provided, however, those engineers may seek indemnification for services arising out of their practice for other than gross negligence, where the engineer's interests cannot otherwise be protected.
- a) Engineers shall conform to state registration laws in the practice of engineering.
  - b) Engineers shall not use association with a non-engineer, a corporation, or partnership as a "cloak" for unethical acts.
- 9) Engineers shall give credit for engineering work to those to whom credit is due, and will recognize the proprietary interests of others.
- a) Engineers shall, whenever possible, name the person or persons who may be individually responsible for designs, inventions, writings, or other accomplishments.
  - b) Engineers using designs supplied by a client recognize that the designs remain the property of the client and may not be duplicated by the engineer for others without express permission.
  - c) Engineers, before undertaking work for others in connection with which the engineer may make improvements, plans, designs, inventions, or other records that may justify copyrights or patents, should enter into a positive agreement regarding ownership.

- d) Engineers' designs, data, records, and notes referring exclusively to an employer's work are the employer's property. The employer should indemnify the engineer for use of the information for any purpose other than the original purpose.
- e) Engineers shall continue their professional development throughout their careers and should keep current in their specialty fields by engaging in professional practice, participating in continuing education courses, reading in the technical literature, and attending professional meetings and seminars.

## 1.10. Additional Resources

- ❖ Wakerley, I. Digital Design. (2006) Prentice Hall
- ❖ Katz, R. Contemporary Logic Design. (2005) Pearson.
- ❖ Lumsdaine, E. Creative Problem Solving and Engineering. (1999) Prentice Hall.
- ❖ Sandige, R. Digital Design Essentials. (2002) Prentice Hall.
- ❖ Nilsson, J. Electrical Circuits. (2004) Pearson.
- ❖ Eide, A. Engineering Fundamentals & Problem Solving. (2002) McGraw Hill
- ❖ MathWorks. MATLAB Reference Material Version R2000a. (2007) MathWorks

## 1.11. Problems

---

1. Use the job listing services online to find five open positions that would interest you and are compatible with the intended degree. Document the title, hiring company, position description and candidate qualifications.

*Note: Be prepared to discuss and present in-class.*

---

2. Identify the top three technology problems that you expect to be solved in the next ten years. Explain your reasons for the selections.

*Note: Be prepared to discuss and present in-class.*

---

3. Name the top three areas of research/trends affecting the future opportunities in computing and Electrical fields. Explain your reasons for the selection.

---

4. Name 5 specialties within Computing and Electrical fields.

---

5. Develop a five-year-plan for yourself and document the plan using the Current-Desired Diagram. For this exercise, show one transition state for each year of the plan.

---

6. Work with your department academic advisor ([engrcs.com/schedule](http://engrcs.com/schedule)) to develop an education plan.

---

7. Associations and clubs are a critical success factors in your education and/or career. Identify five clubs or associations that you feel would best support your education and/or career. Next, select one as the most important one and explain your reason for the selection.

---

8. What are patent claims? Explain their significance and provide a claim example.

---

9. Develop a patent application for one of your new ideas. A new idea is defined as a feature, product, service or improvement that is currently not available on the market. Your report is expected to include the following:

- 1) Problem definition
  - 2) Description of similar products on the market that may solve the problem, and comparative advantages/disadvantages of your product idea.
  - 3) Include the following information:
    - a) Claims (specific inventions)
    - b) Research to support your invention
    - c) An instantiation of the Claims (product utilizing the invention)
    - d) List of patent numbers and titles for the top three relevant patents
- 

10. What are the six fundamental canons of NSPE code of ethics?

---

11. In regard to Code of Ethics, what is definition of a stakeholder and which groups can be considered stakeholders?

---

12. Your company bids on a government job and gets it. You are an engineer at this company. Further, you know your company is making above market profit on the government contract. What is your ethical responsibility and why? Base your answer on specific canons of NSPE Code of Ethics.

---

13. You are a project engineer for a heart monitor development project. You have been reviewing the test data that shows 2 out of 100,000 tests failed. Your manager thinks .002% is a great test result and the product should be released to market. What is your ethical responsibility and why? Base your answer on specific canons of NSPE Code of Ethics.

---

---

14. Your company has a contract to build a bridge. Material specified in the contract is no longer readily available, so your company substitutes another type of material. What are your disclosure responsibilities at this point and to whom? Base your answer on specific canons of NSPE Code of Ethics.

## **Chapter 2. Teamwork and Communication**

### **2.1. Key Concepts and Overview**

- Thinking Modes or Styles
- Teamwork
- Communication



## 2.2. Thinking Modes

Thinking modes, also referred to as the mental models, thinking preferences or thinking styles, is the foundation by which we make decisions, plan, communicate and execute tasks. Organizations with employees who have shared thinking styles are the most effective organizations since they have improved communication and teamwork. Their effectiveness results from the fact that a shared model is the process we use to understand our environment and decision making processes.

Although there are a large variety of evaluation tools in the market for thinking modes evaluation, this text introduces a variation of the Herrmann Brain Dominance Model (HBDM). These types of evaluation are most commonly referred to as “personality tests” and are often used by employers to assess teamwork, job performance and communication preference of their employees. One of the most common personality tests used in business is called the **Myers-Briggs personality test**.

The reason for the presentation of the HBDM here is to provide a tool for evaluating self and teammate preferences in order to improve communication resulting in improved teamwork and performance. Another benefit is to have sufficient information to create a team with diverse ways of thinking. Since it has been shown that diverse teams outperform less diverse-thinking teams.

The HBDM attempts to identify an individual’s preferred thinking mode in accordance with four thinking styles (also called thinking modes preferences). The four thinking modes are:

### **Analyzer (A)**

- Requires facts and data that fits together logically
- Likes data analysis
- Learns and makes decision based on information from experts and proven facts

### **Manager (M)**

- Requires a proven and detailed execution plan that minimizes risks
- Likes low-risk activities with clear step-by-step plan from beginning to end
- Learns and makes decisions based on testing results and proven experience

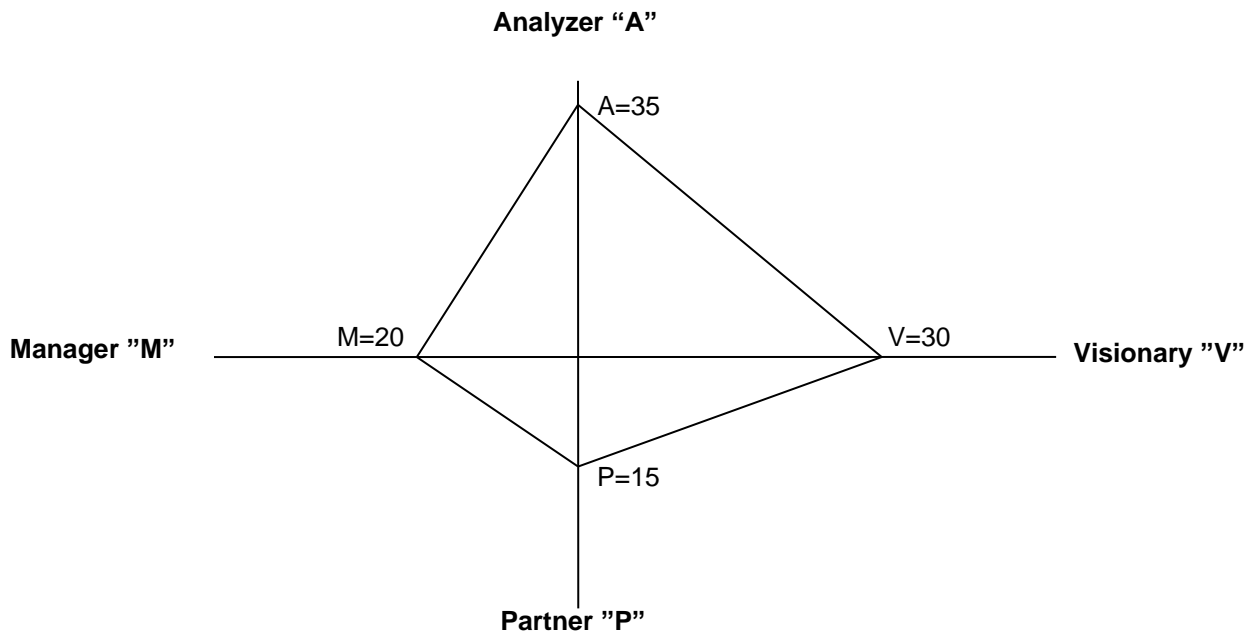
### **Partner (P)**

- Requires human interaction and knowledge that everyone is taken care
- Likes to work on relationship development and help with environment/community/politics
- Learns and makes decision based on feeling, personal experiences and in-person feedback from others

### **Visionary (V)**

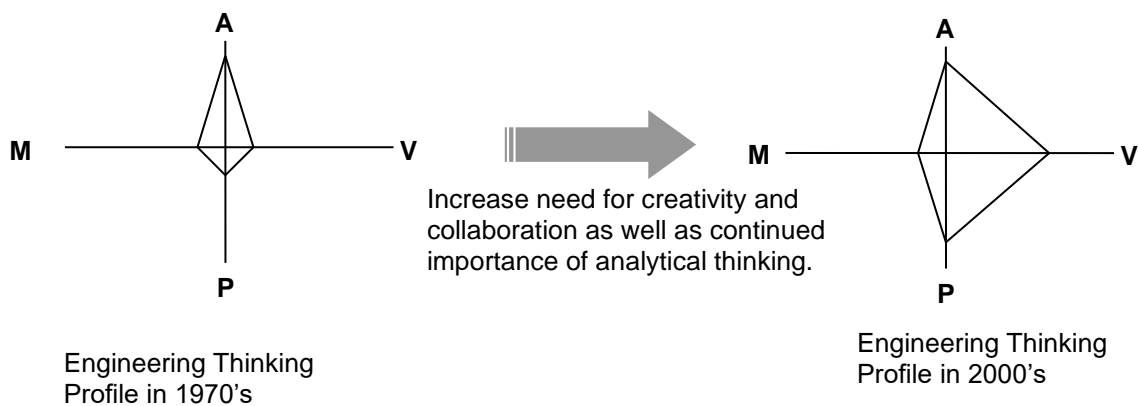
- Requires change and innovation and loses interest once the project is understood
- Likes to work based on inspirations and insights
- Learns by conceptualizing possibilities and exploring alternative to known solutions and processes.

An individual may be stronger in one or more of the modes, but most humans move among the different thinking modes. Typically, the assessment is displayed as a radar chart. For example, someone receives scores of A=35, M=20, P=15, and V=30, then the evaluation may be plotted as the following radar chart:

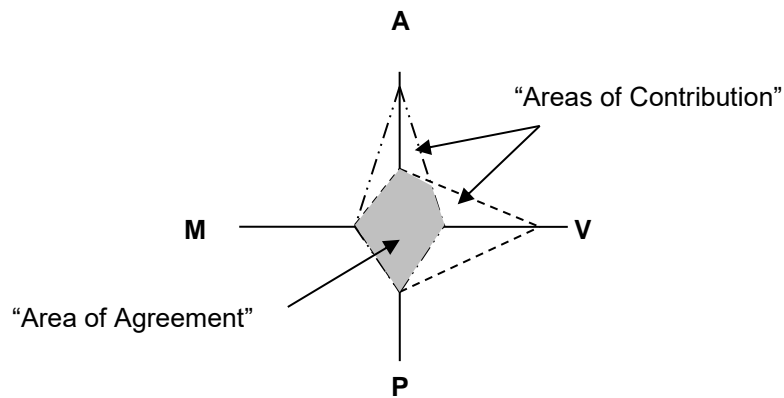


Based on the above chart, we can state that this individual is highly analytical (A=35) and creative (V=30), but not as interested in managing or working with others. Thus, it is important that other teammates have stronger Partner and Manager attributes to compensate, for optimal performance.

Although such an individual would have had great success as an engineer in the past, in today's collaborative environment, engineers need stronger collaboration skills. Over a 40-year period, the thinking attributes of a successful engineer has changed to require more creativity and collaborative skills:



Thinking model is an important consideration when selecting team members, since it allows for a diverse team to be put together. The following diagram shows the overlap of two persons' thinking profiles:



The areas of overlap is called the “Area of Agreement”. As the name implies, when both people are thinking in the Area of Agreement, they will reach the same conclusion given the same facts. Although some overlap is good and improves communication, it does not contribute to new idea generation.

The area of non-overlap is called “Areas of contribution” since each person will have a different perspective and is likely to have a different solution given the same facts. Therefore, teams operating in this region have a higher probability of generating creative solutions with broader effectiveness. The drawback is that there are more potential conflicts.

When selecting team members, it is important to balance the Area of Agreement with the Areas of Contribution. One allows for improved teamwork while the other allows for increased creativity and diversity in thinking.

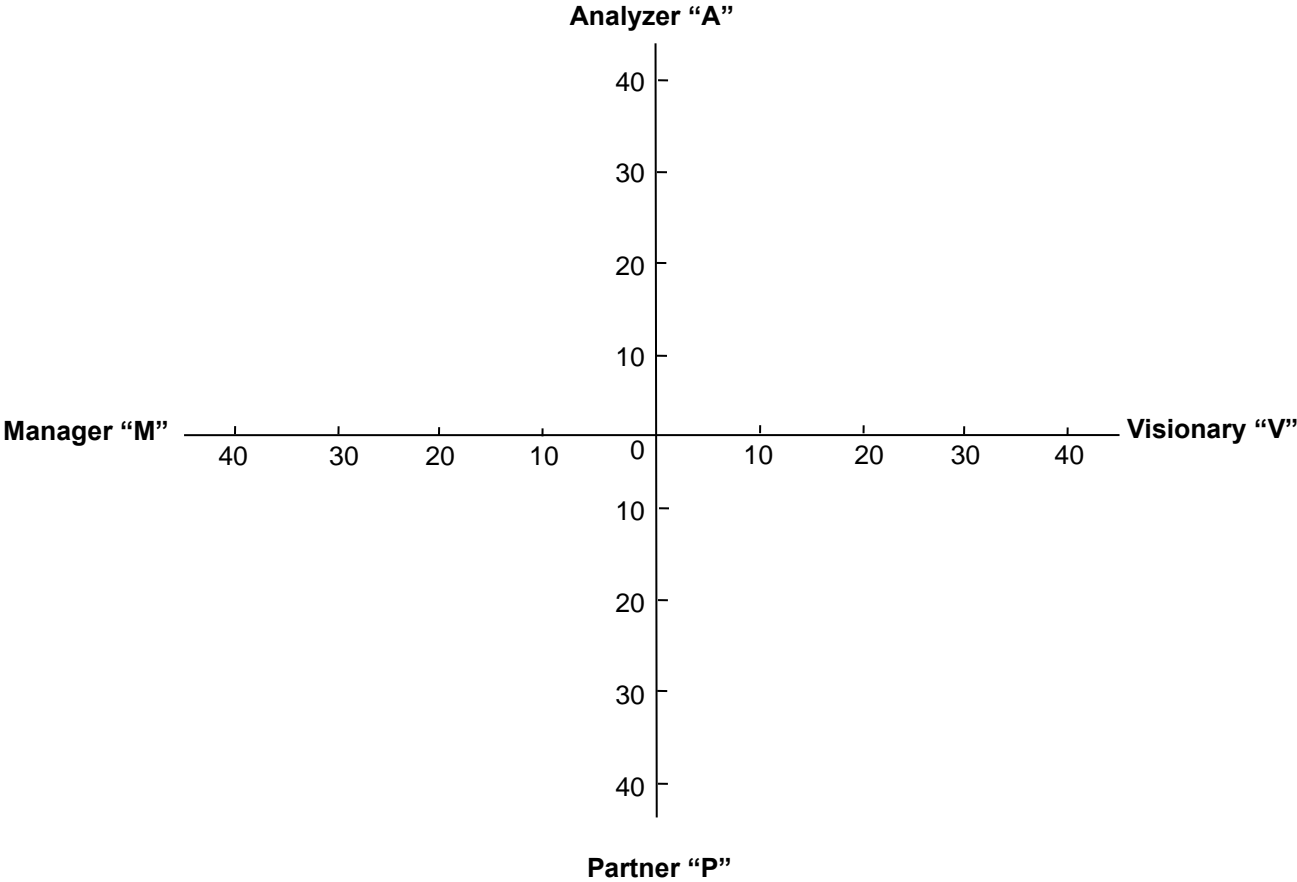
### 2.3. Thinking Mode Assessment Based on Herrmann Brain Dominance Model (HBDM)

The following assessment tool is designed to assess the thinking preference of the user by completing the tool and drawing the resulting radar chart that may be analyzed. This tool is adapted from a pre-existing tool (unknown original source).

**STEP 1. For each statement row, rank response columns from 1(least likely) to 4 (most likely).**

Statements	A Rank	M Rank	P Rank	V Rank
1. In a project, you prefer to: A. Work with data and facts; M. Develop a plan and schedule; P. Meet and discuss ideas with others; V. Look for the broader view and next new idea.	.....	.....	.....	.....
2. You best learn new ideas by: A. Applying them to actual situations; M. Conducting careful analysis; P. Discussing with others; V. Thinking of new ways to use the idea.	.....	.....	.....	.....
3. When working on a problem, you: A. Organize the material logically but not too much detail; M. Organize material neatly and pay attention to detail. P. Look for impacts on others and society; V. Explore new ways to solve the problem.	.....	.....	.....	.....
4. In studying new material, you prefer to: A. Read the textbook and listen to fact-based lecture. M. Test processes and procedures to find the problems; P. Read the introduction and overview to understand the purpose; V. Explore possibilities by asking 'what-if' questions.	.....	.....	.....	.....
5. After solving a problem, you: A. Think through ideas rationally; M. Find practical uses of the knowledge learned; P. share with others; V. Appreciate the elegance of the solution and explore other solutions.	.....	.....	.....	.....
6. The most important item in a new article for you is: A. Its basis in fact; M. Validity of the idea based on past data; P. Practicality of the idea; V. The relevance of the idea to your opinion.	.....	.....	.....	.....
7. In project execution, you prefer to: A. Make up a theory and then test it; M. Develop a plan and execute the project according to plan; P. Do experiments and observe the results; V. Experiment and explore ideas and possibilities.	.....	.....	.....	.....
8. You are most influenced by: A. Ideas that are based on facts and logic; M. The conviction and force used to express the idea; P. closeness of idea to your opinion; V. Ideas that are broad in nature and forward looking.	.....	.....	.....	.....
9. In your decision-making process, you rely most on: A. Reality and facts; M. Proven track record, detailed studies; P. Discussion with others and their opinions; V. Your own feeling and views.	.....	.....	.....	.....
<b>Total:</b>	.....	.....	.....	.....

**STEP 2. Plot your scores on the following Radar Chart:**



**STEP 3. Describe the Chart in Terms of Thinking Preferences**

## 2.4. Teamwork

In the past, it was possible for engineers to work in isolation with little or no interaction with peers, teams, departments, and organizations. Increasingly, engineers work as a team in order to accomplish their tasks. In a today's typical project team, members include all business functions (marketing, design, manufacturing, customer support, finance, risk management, regulatory), customers and suppliers.

Success as an engineer relies heavily on the engineer's ability to work effectively in a diverse team. In this section, we will discuss team processes and tools that improve team effectiveness.

### Steps to organize an effective team:

- 1) Select members who complement each other's Thinking Model (Herrmann Brain Model) and project expertise.
- 2) Develop a shared vision as a team. Team members should be visualizing project success similarly.
- 3) Agree on a shared objective, including measures of success and a delivery timeline.
- 4) Generate a shared execution plan that is consistent with the project's vision and objectives. The plan must be agreed upon by all members. The plan should include:
  - (i) Roles and responsibility
  - (ii) Structure
  - (iii) Deliverable timeline
- 5) Evaluate the team performance based on the results. Modify vision, objectives and execution plan to improve performance.

### Student Exercise – Team

Identify 3-4 other students to form a team by working with your instructor and use the above steps to organize an effective team.

### Solution

#### “Team members” of a typical engineering project include:

- Customers
- Sales and Marketing
- Design, Development (all disciplines)
- Testing
- Implementation and Manufacturing
- Quality Control
- Regulatory Agencies
- Risk Management
- Investors

## **Team Development Overview**

Team development has been the subject of numerous studies and go back to early days of human development. In recent times, it is understood that teams develop and mature through a well-defined four stages development process.

Although change in team members, environment or project definition may result in the team moving from one stage to the next and back. The four stages of team development are Forming, Storming, Norming, and Performing. This 4-stage development process was first proposed by Bruce Tuckman in 1965. He believed that all the four stages are necessary and inevitable in team development and its ability to work on projects as a team.

It is important to know that not all teams reach the performing stage; most teams move up and down the development stages depending on the project challenges and current environment. Below is a more detailed description of the Four Team Development Stages:

### **Forming Stage**

This is the first stage of team development where team members are attempting to learn about each other. Members work on getting to know some basic information about each other such as contact information, experience, skill, and interests. Teams in this stage also attempt to understand or establish acceptable group behavior, purpose, and goals.

### **Storming Stage**

Teams move to Storming from Forming Stage. This transition occurs after team members have learned about each other and now are attempting to test the limit of each other's ability and limitation. This testing process typically leads to more conflicts. For a successful team, this stage will allow the members to learn to deal with conflicts.

### **Norming Stage**

Upon completion of Storming Stage, team typically moves to the Norming Stage. In this stage, team members already know each other, have established limits and are able to resolve conflicts. For many teams, Norming is the highest level the team will attain. Teams in this stage are developing and adopting team processes that are mutually acceptable and are able to effectively deliver results.

### **Performing Stage**

Although teams in Norming Stage are delivering results and are effective, it is the rare team that reaches the Performing Stage and can deliver results at levels that are significantly higher than the Norming Stage team. Members of a team at the Performing Stage have accepted each other's strengths and weaknesses. Further, they can effortlessly build on each other's strengths and supplement each other's weaknesses. A team in Performing Stage delivers spectacular results with seeming little effort - much like a perfectly tuned machine.

### **Student Exercise – Team Status**

Work with your Team to identify your team's current stage of development.

### **Solution**

## **Tools for Managing Teams**

The following tools have proven to be useful in improving the effectiveness of teams. It is recommended

that early in your team development, review the list and identify the tools your team will use in the project.

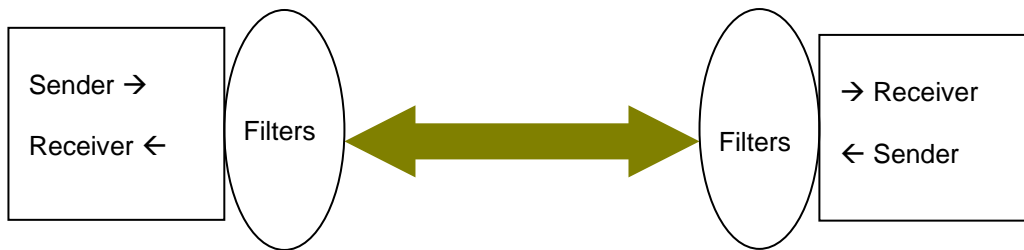
- Team's mission statement and objectives
- Project plan and timeline
- Team member role definitions
- Team ground rules
- Meeting agendas
- Task and issues list
- Team member evaluations (Peers, supervisor, subordinates)



## 2.5. Communication

In all communications, there is a sender and a receiver of information. In successful communications, the impact of information on the receiver is the same as is expected by sender of the information.

In human communications, understanding of information is colored by the biases of the receiver and sender, shown as filters in the following diagram. Additionally, the information being transmitted includes the formal part (verbal or written) and informal part (body language and other informal indicators conveying feelings, values, hopes and dreams). Therefore, communication needs to be focused and adapted to the audience for success.



The single biggest challenge in communication is the assumption that communication has taken place. All effective communicators pay special attention to clues of validity of this assumption.

As mentioned earlier, knowing the audience is the key to effective communication. Fortunately, the Herrmann Brain Model is an excellent tool for knowing audience preferences. The following table provides a guide of best approach for the various thinking preferences:

<p><b>Analyzer “A”</b>  <b>”Needs Facts and Logic”</b></p> <ul style="list-style-type: none"> <li>• Facts and Data are supported and documented</li> <li>• Communication is supported by logical use of facts and data</li> </ul>
<p><b>Manager “M”</b>  <b>”Needs Predictability and Low Risk”</b></p> <ul style="list-style-type: none"> <li>• Plan is in detail and has been tested and proven by past experience</li> <li>• Communication is well-organized, detailed and meets the requirements</li> </ul>
<p><b>Visionary “V”</b>  <b>”Needs new challenges and change”</b></p> <ul style="list-style-type: none"> <li>• Plan is new, exciting, outcome is unknown and has not been tried before</li> <li>• Communication must be imaginative, futuristic and emphasize the change/newness</li> </ul>
<p><b>Partner “P”</b>  <b>“Needs relationship and emotion appeal”</b></p> <ul style="list-style-type: none"> <li>• Plan has to be people centric, improve relationships and society</li> <li>• Communication must be playing to emotions and people experiences</li> </ul>

## **Student Exercise – Communication**

Meet with your team and discuss the team members and discuss:

- Thinking preferences
- Technical skills in Electrical, Computing and Mechanical Assembly
- Communication, Teamwork, and other relevant skills

At the end of the first meeting, each member should answer the following questions:

- All members' first and last names
- All members' strongest skills
- Gaps in the team
- Team name

## **Solution**

There are a variety of communication situations, and each requires its own approach to communication. Here are the most common communication categories:

- Negotiation
- Group Presentation
- Written Reports
- General Social or Business Conversation
- Remote (Phone, Online, ...)
- Student Exercise - Identify other communication categories?

In the remainder of this section, we will focus on the Negotiation, Group Presentation and Written Reports.

### **Negotiation**

Negotiation is a form of communication that everyone engages in during life and work activities. In general, there are three types of negotiation based upon the power relationships between parties.

#### **Soft Negotiation**

Power Position: One side is giving in; usually due to real or perceived lack of power.

Results: Commonly leads to resentment in the long-term and poisons future relationships.

#### **Hard Negotiation**

Power Position: Both parties either have or are perceived to have equal power and typically are focused on the short-term where neither party gives in.

Results: Hard negotiation leads to conflict and counter-productive behaviors but typically exposes areas of conflict.

#### **Principled, interest-based or Win-Win Negotiation (ideal)**

Power Positions: Both parties approach the negotiation as equals with interest in results that maximize total benefits.

Results: These types of negotiations require a high level of trust and time to complete. All issues are decided on merit and to maximize benefits to both parties. This fact leads to both parties working

toward a superior outcome that will benefit everyone concerned. Principled Negotiation benefits both parties at a higher level in the long-term as compared to other types of negotiations.

### **Group Presentation**

Group Presentation is one area of communication that most professionals do not engage in as often as other forms of communication. But it is still important for a successful education and career. Here are some key considerations in group presentations:

- Know your audience's interests and issues. Create your presentation with the audience in mind.
- Command attention and get the audience on your side.
  - Introduce yourself
  - State the purpose of the presentation in terms that capture the interest of the audience
  - Acknowledge the needs and expectations of the audience
- Audience will be looking for non-verbal cues from the moment you start.
  - Stand erect
  - Seek eye contact
  - Speak clearly with calm authority
  - Project energy, enthusiasm and competence
  - Use visual aids only to emphasize key points
- Audience can remember 3 to 5 main points only if they are repeated and reinforced
  - Focus on 3-5 main points
  - Keep reinforcing key points in your presentation, visual aids and printed materials
- Be aware of your audience's dominant thinking mode and learning style (Herrmann Brain Model)
- Your audience has a busy schedule
  - Keep to your timeline
  - Practice your presentation and make sure that it is designed for the time allowed
  - Manage questions to ensure you can cover your main points
  - Make sure you have enough time for your conclusion and summary

### **Written Technical Report**

Technical reports are not essays or stories; the report is intended to communicate key information in an easy to understand and search format. Reports must be clear as to the problem being solved, research, analysis, recommendation and outstanding issues. An effective engineering report should contain the following sections, with clear titles and separation for each section:

- Cover page includes report title, author, date plus a brief (1-2 paragraphs) executive summary (Abstract) of the problem and recommendation only.
- Problem definition includes statement of problem, scope, schedule, and resource.
- Research section includes publication research, web research, experimentation, and experts in the field.
- Analysis applies the research results in solving the problem.
- Recommendation / Conclusion  
The optimal decisions, implementations or solutions that are being recommended. It is

important that the reader reaches similar conclusions upon reading earlier sections.

- **Issues / Dependencies**  
Issues or dependencies that have been encountered but have not been resolved as part of the work being reported.

## 2.6. Additional Resources

- ❖ Eide, A. Engineering Fundamentals & Problem Solving. (2002) McGraw Hill
- ❖ MathWorks. MATLAB Reference Material Version R2000a. (2007) MathWorks
- ❖ Spiegel, I. & Torres, C. Manager's Official Guide to Team Working (1994) Wiley

## 2.7. Problems

- 
1. Complete a self-evaluation using the HBDM Form. Include the resulting 4-category scores with a description of what each score implies and the radar chart.

---

  2. What are the descriptions of Area of Agreement and Area of Contribution in the context of HBDM?

---

  3. Do you think it is a good to have everyone on the team have identical thinking model? Explain the reasons for your answer.

---

  4. What are the four stages of Team Formations? Which do you believe is the most productive stage?

---

  5. What are the four most important tools for managing teams?

---

  6. When presenting to an Analyzer what should you emphasize in your communication?

---

  7. When presenting to a Visionary what should you emphasize in your communication?

---

  8. What are the three negotiation types?

---

  9. What is your latest experience with negotiation? Explain the situation, and also identify the type of negotiation used. In retrospective, was it the best type of negotiation to be used for that situation?

---

  10. What are the top five considerations when presenting to a group?

---

  11. Why do engineers need to be skilled in communication and teamwork? Support your answer with an example.

---

  12. What are the four most important sections of written technical reports?

## **Chapter 3. Creative Problem Solving**

### **3.1. Key Concepts and Overview**

Step 1. Customer Issues/Needs Identification

Step 2. Problem Definition

Step 3. Idea Generation

Step 4. Optimal Solution Selection

Step 5. Solution Implementation

### **3.2. Creative Problem-Solving Process**

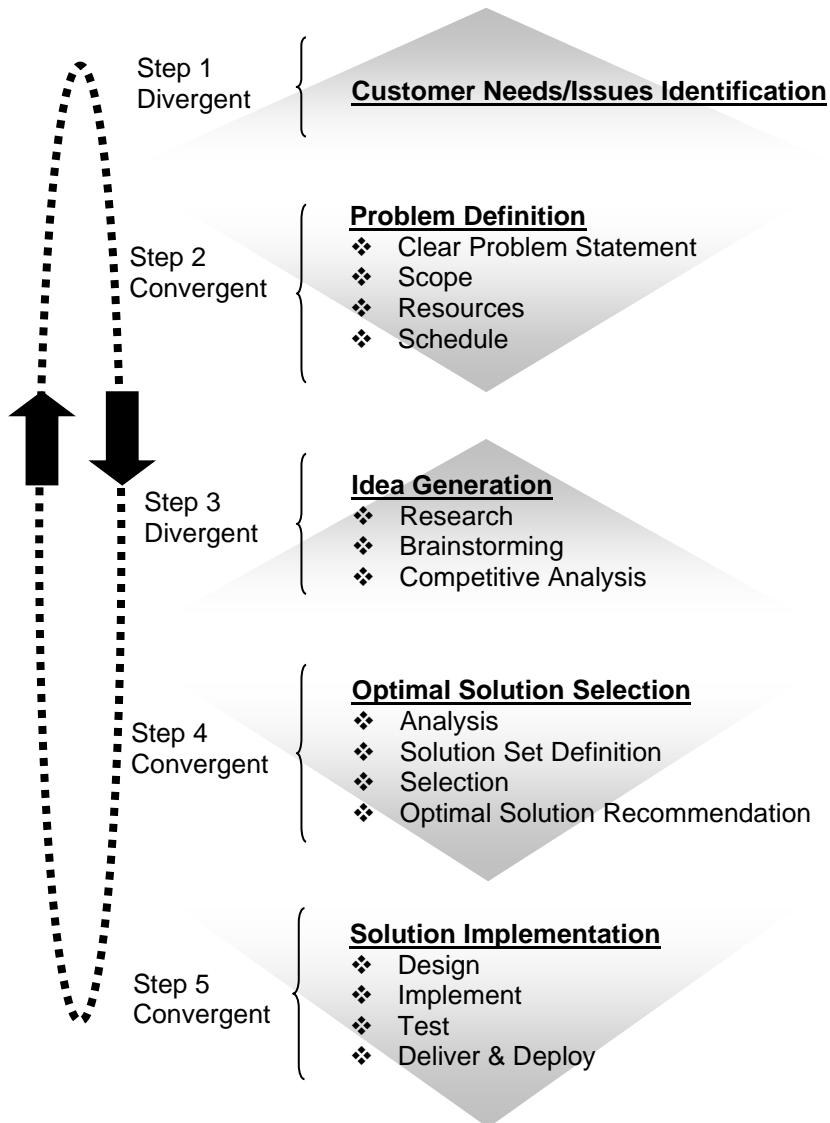
Creative problem-solving is the core of any engineering position in today's world. There are many approaches to creative problem-solving. This chapter introduces creative problem solving in 5 steps as shown below:

1. Customer Issues/Needs Identification
2. Problem Definition
3. Idea Generation
4. Optimal Solution Selection
5. Solution Implementation

Although steps are shown consecutively, creative problem solving is an iterative process; that is, results from latter steps may impact earlier steps in the process. In these situations, the earlier steps must be modified, and all latter steps must be reworked.



## 5-STEP CREATIVE PROBLEM SOLVING PROCESS



### **Notes**

- 1) Creative Problem Solving process is iterative with many feedback loops.
- 2) Convergent Thinking refers to the process of reducing the number of possibilities through a selection process.
- 3) Divergent Thinking refers to a process that expands the number of possibilities through a broadening of the scope and removing limitations.

### 3.3. Step 1 - Customer Issues/Needs Identification

Successful companies and engineers solve problems that add values for the customer. So in all cases, there exists one or more customers who either have unmet needs or issues that need to be addressed.

Typically, engineers work with marketing specialists and others to collect relevant information and compile them into a list which will be used to identify the problem.

The process to document customer issues and needs may include one or more of the following:

- Personal experience - If your organization is experiencing similar problem then you should work with your organization to collect first-hand experience and understanding of the problem.
- Customer surveys and reports - Customer surveys and reports can provide a listing of unmet needs or issues.
- Customer Support - The Customer Support team has identified a common issue or need with current product or service that should be addressed.
- Competitive Analysis - Competitive analysis is the process that helps identify the relative strengths and weaknesses with respect to competitor's product. This information may be used to define the problem.
- Trends - Market and/or technology trends among forecasting of future issues, unmet needs, or opportunities.

Commonly, there are multiple sources, and the list may include hundreds of items.

#### **Student Exercise – Needs Analysis**

As a team, identify issues/needs that you believe if resolved, would add the highest value to your organization and/or community.

#### **Solution**

### 3.4. Step 2 - Problem Definition

This is the most important step of creative problem solving, since only solving the right problem adds value. Regardless of the excellence of the final solution, if one is not addressing the right problem, then it is a waste of time and resources.

A complete problem definition has four distinct components:

- Problem Statement
- Scope
- Resources
- Schedule

#### Problem Definition Process

- 1) Agree that there is a problem worth solving.
- 2) Look for the root cause, not just the observable symptoms.
- 3) Look at the trends in order to understand the problem's magnitude. Studying the trends also helps in understanding the development of problems in a wider context and timeframe.
- 4) Research information in the problem areas. The goal is to learn from others who may have similar problems.
- 5) Look at the problem in the context of its surroundings, and past and future trends. This approach is also called System Thinking where the problem is defined for the whole system not just one part of the system.
- 6) Write the Problem Definition in term of its four components:
  - Problem Statement
  - Scope
  - Resources
  - Schedule

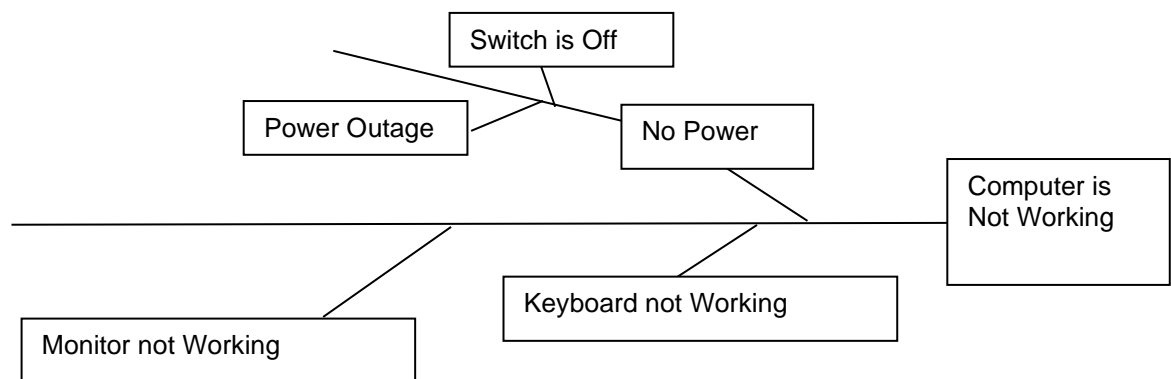
#### Methods to find the root cause

Understanding and solving the root cause is important, since solving a symptom is at best a temporary solution and at worst is no solution. This section describes the most common tools for identifying the root cause:

- 1) **Ask questions** (Kepner-Tregoe "KT" approach)
  - Who, What, Where, Why, how much, etc. questions?
  - It is a good technique for finding the problem boundaries
  - KT approach also identifies requirements that are outside of the scope of your problem and should not be included in the solution considerations.
- 2) **Survey**

Manufacturers and services companies use surveys to collect customer experience data. You may see this type of survey being called the "Voice of Customer", or other descriptive names. This data is usually presented in a Pareto chart (listing each problem and number of occurrences) which helps in deciding what problem to work on.

- 3) **Statistical Process Control (SPC)**  
The SPC sets a limit for each parameter. When one parameter is out of the set range, then that item is identified as a problem to be worked on.
- 4) **Failure Mode and Effects Analysis (FMEA)**  
Explore all possible failure modes of the product or process, and their effects on the result (customer experience).
- 5) **Failure Tree Analysis (FTA)**  
Graphical view of the possible element failures that would result on a specific system failure.
- 6) **Fishbone Diagram**  
Use a cause/effect diagram leading to the problem. Typically start with the problem as the head of the fish, and the possible causes of the problems are drawn as bones.



- 7) **Experiments and Weibull Analysis**  
Carefully constructed experiments to answer a specific list of questions or to collect a specific set of data.
- 8) **Benchmarking**  
Typically, use a world class competitor as the benchmark to see if your product or process has a comparative problem.
- 9) **Introspection**  
When time is too short to do an in-depth data collection and analysis, one can think about what one knows and decide. Some say we do not spend enough time in this mode, while spending too much time asking others or collecting data that is never analyzed.

### 3.5. Step 3 - Idea Generation

In this Section, the objective is to come up with as many ideas to solve the problem as is possible. The two steps in this process are research and brainstorming.

It is important to understand the current state of knowledge with the respect to the problem being solved. So the first step is to access all sources and to compile all the relevant material:

Some common sources of information include:

- Networking - take advantage of meeting others and learning from their experiences.
- Searching the web
- Keeping an idea file
- Modeling a problem
- Searching relevant patents
- Experimenting
- Studying the Competitive solutions

Once all team members understand the problem and knowledge of existing relevant research data, then it is time to brainstorm possible solutions.

#### Brainstorming Rules

Although brainstorming is a creative process with as few boundaries as possible, in order to maximize the creative ideas, it is still important to have rules for an effective brainstorming. The four rules of brainstorming are as follows:

- **No criticism is allowed** – defer judgment until later. This is the most important rule!
- **Generate as many solutions as possible** – quantity counts, so use few words.
- **Wild ideas are welcome** – be as creative as possible and do not worry about practicality of your ideas.
- **Leveraging is encouraged** – build on the ideas of others

Brainstorming may be verbal or written, and is outlined here:

#### The Verbal (classic) Brainstorming Method

The key to successful brainstorming session is to be prepared, follow a procedure and debrief. Here are some points to consider:

##### Preparation

- **Team member**
  - (1) Select several members with quadrant V preferences
  - (2) Have a direct or indirect representation from stakeholders (customers, design, test...)
  - (3) The best number of people is 3-10.
- **Location**
  - (1) People think more creatively in unfamiliar locations, so hold the brainstorming off-site.
  - (2) People should be seated so they can see each other and can easily hear each other
- **Scheduling**
  - (1) Earlier in the day is better, since brainstorming is exhausting work.
- **Materials**

Have the necessary equipment available to stimulate creativity: easel, flip charts, markers, note cards, visual aids, and props.

##### Procedure

- **Briefing**  
Allow team members a few minutes for social interaction and for each person to comfortably stake out a personal space in the seating arrangement.
- Review the 4 rules of brainstorming.
- Explain the procedure.
- Do warm up exercises.
- **Brainstorming**
  - (1) Ensure that the problem statement is well understood
  - (2) Ask team members to start sharing ideas. Start by bringing out the obvious in order to get it out of their minds and move forward to more creative solutions.
- **Close**  
Once the idea generation has slowed down and time is running out; give a 5-minute warning. Typically, great ideas come out in the last 5 minutes.
- **Dismissal**  
Collect all notes and ideas for later evaluation. Encourage team members to send you any ideas that come to them after the meeting.

### Debriefing

The best way to improve your brainstorming process is to find out what worked well and what did not.

### Written Brainstorming Methods

Written brainstorming methods help in situations with large numbers of participants or shy participants. For success, it is important that the four rules of brainstorming are followed and that the ideas are written down quickly with no regard to correctness of format or content.

- **Pin Card Method**  
The Pin Card Method is applied by seating the participants around a large table and having them write down ideas on note cards – one idea per card. The note cards are passed, and each member is asked to add his/her improvement or related idea to the card.

**Student Exercise** - Apply the Pin Card Method to ideas on improving this class. Start with a sheet of paper for each row and pass it down and back. Each person on each pass must add one item to the list.

- **Crawford Slip Writing Method**  
The Crawford Slip Writing is used for large group brainstorming. Once the problem is defined, and each participant is asked to write down 20-30 ideas on slips of paper. Each idea should be written on a different slip of paper which is then collected quickly.
- **Anonymous Method**  
The anonymous method features minimal face-to-face interaction. An idea is written on paper and submitted to others in the organization. Each person can modify the idea or add their own. The important feature of this method is that the author of the idea remains anonymous.
- **Panel Method**  
The Panel Method is used for very large groups, and is typically conducted according to the following steps:
  - 5-7 volunteers are chosen from the entire group to be on the panel.
  - The problem is presented to the whole group.
  - The panel verbally brainstorms for 20 minutes, posting their ideas on flip charts.

- The rest of the group is able to add their new ideas or improvements to the list after the panel is done.

**Electronic Brainstorming**

The Electronic Brainstorming uses online forums, bulletin boards or other electronic community formats in order to capture brainstorming ideas. Electronic brainstorming is used for large groups who are widely distributed through time, geography, or other dimensions.

### 3.6. Step 4 - Optimal Solution Selection

As the name implies, this section outlines the process to select the best or optimal solution. The Optimal Solution Selection is accomplished via analysis and solution set definition and optimal solution identification; that is to say, all possible solutions must be recognized before the optimal solution can be determined.

Similarly, to brainstorming, optimal solution selection also has four rules:

- Look for quality and better ideas
- Make “wild” ideas more practical
- Synthesize ideas to obtain more complete, optimized solutions
- Maintain a positive attitude; continue to look for improving solution ideas

The first step of the process is Analysis and Solution Set Definition, which can be accomplished in the following three steps:

#### **Step 1: Sorting related ideas into categories**

- Lay out the ideas from the brainstorming on post-it notes or index cards (one idea per card/post-it note).
- Allow team members to look at the cards for a period of time.
- Ask them to put cards with similar ideas together.
- Add category title cards on each grouping of similar ideas as they form. Use either color cards or symbols that have nothing to do with the area of discussion; you do not want to create any presumption of category definition.
- If an idea fits into more than one group, then create a duplicate card.
- If an idea does not fit in any existing category groups, then create a new category.
- You should keep the final number of categories to 5-7. If there are more, repeat the process to see if any can be combined.

#### **Step 2: Developing quality ideas within a category**

- Work on one category at a time. If you have a large team (more than 7), then create multiple teams and have each team work on a separate category at a different table or location.
- Let each team know that the goal of this step is to “engineer” the many ideas within the category down to fewer, and more completely developed, practical and higher-quality ideas. This is called “Idea Synthesis”.
- The combined idea is written on the same or a new card, but make sure to keep the original idea attached to the new synthesized idea. Follow the process until the category is completed.
- Danger: The team gets focused on one novel idea and ignores the rest.

#### **Step 3: Force-fitting unrelated ideas between categories**

- This is where the whole team comes together and tries to combine ideas from all the categories in order to come up with a superior solution.
- In some situations, it is acceptable to delay the process until a later date if the situation dictates it.

Once the analysis and solution set definition process in the last three steps has been completed, you may have 3-5 more practical and better-defined solutions. At this point, you are ready to work on Optimal Solution Identification from this smaller set so use the following three steps in order to pick the solution to recommend and implement:

- 1) Establish evaluation criteria; list of solution attributes that are needed.
- 2) Rank ideas/solutions based on the evaluation criteria.
- 3) Decide on the best solution or idea to be implemented.



Now, let's discuss details involved with each of the steps:

### Step 1. Selection Criteria

A good list of selection criteria includes all factors that influence a problem or decision. In choosing the Selection Criteria, consider the following factors:

- **Balanced list** - analytical and intuitive criteria
- **Boundaries** - scope, schedule, and resource limitations
- **Projection** - look to the future and consider factors that would simplify the implementation and make the sell and support process more effective among other factors.

### Step 2. Ranking Ideas

Rarely one idea emerges that will meet all of the selection criteria, so typically, there is a need to think of ways to rank possible solutions. The approach you pick will depend on the timeline, complexity and importance of the decision.

- **Pure Voting**
  - Each person votes based on his/her understanding of optimal solutions.
  - Issue: Any potential benefit that may come from discussion is lost and may also be negatively affected by peer-pressure/group thinking.
- **Voting Variations**
  - **Agreement gradient**  
Allow each person to vote from 1 (cannot live with the idea) to 10 (the best idea ever). This method enables the participants to discuss their choices.
  - **Ranking**  
Instead of picking one idea, each person ranks ideas in order of preference and discusses their choice. Many companies use this method for selecting employees.
- **Advantage/Disadvantage Technique**  
The most useful way is to add weighting factors to each criterion and then compare how each idea ranks vs. the criteria. Two common weighting approaches used are:
  - + for positive, - for negative and 0 for no impact
  - -10 for large negative impact and + 10 for large positive impact

Here is an example of a Selection Criteria Table used to document the ranking of different degrees using the -10 to +10, each criteria weighted (note you can get as detailed as you need to be):

Criteria (Weight)	Ideas / Potential Solutions / Options		
	Comp. Sc.	Comp. Eng.	Elect. Eng.
Criteria 1 (x20)			
Criteria 2 (x50)			
Criteria 3 (x30)			
...			

- “\$100 decision-making” is a variation of ranking process where you have \$100 and can spend it on the criteria according to importance.
- **Advocacy Method**  
The Advocacy Method works best for a small number of ideas. Each team member is assigned one idea to defend in front of the team. The team makes the final decision based on the

presentations.

- **Reverse Brainstorming Method**

In the Reverse Brainstorming Method, each member is assigned one idea to criticize its weaknesses and flaws. After each member explains their criticism of their idea, the team chooses the best one.

- **Experimentation or Taguchi Method**

The Experimentation/Taguchi Method relies on well-designed experiments and trial-and-error methods in order to decide on the best solutions.

The best (extreme) example of this approach is the work on the light bulb. Thomas Edison and his lab used over 1,000 different materials to identify the material that would work best as a filament in a light bulb.

### **Step 3. Decision Making**

The ranking methods do not always come up with a clear method, so here are some common ways decisions are made:

- **Coin Toss**

If ideas are equally good, then coin toss (random selection) may be the best answer.

- **Easy Way Out**

If ideas are equal in merit, then implementation considerations may be used. Select the final solution based on the ease of implementations.

- **Refine Criteria**

If the first set of criteria did not result in a selection of the solutions, then modify existing criteria or add new criteria to better separate various solutions. The updated Criteria may show advantages and disadvantages that might have been missed the first time through the selection process.

- **Consensus**

If time is short, you may want to try for Consensus, which means select the idea that most can agree to. The result of Consensus process is typically poor, because participants go for the easy and obvious ideas, avoiding creative or new ideas.

- **Hybrid Decision**

If there is sufficient time, a Hybrid Decision Method can be used. This method synthesizes the best solutions by integrating the best of each idea. This is the highest form of group decision-making, and it is a highly recommended approach for creative problem solving.

- **Compromise**

The Compromise Method reaches a decision through trade-offs among the top selected solutions. This form of decision-making is commonly used in government and is not recommended for creative problem solving.

- **No Decision or Delay**

Sometimes it is wise to delay and not make a decision just as long as it is recognized that no decision or delay is also a decision. If the decision is delayed in order to collect more information, and not because no one wants to take a risk, then it should be considered. In all other cases this would be problematic.

- **Intuitive**

For many people, intuitive decision making is very effective. It is common to come up with a decision intuitively and then rationalize it. Although this approach is not commonly accepted, it is important to give appropriate weight to intuition in final decision-making.

### **Decision Analysis**

Now that the decision is made, how can one be sure that the decision is correct? This is a very important question that should be asked every time a decision is made. It can help find flaws in the decision and also help to improve future decision-making processes.

Here is a checklist of activities to consider:

- Can the idea be combined to obtain a higher-quality solution?
- Can different ideas of equal quality be implemented all at once?
- How well does the solution or idea solve the problem or meet the problem definition?
- Validate that the ideas meet all stated needs.
- Complete a risk analysis on implementing the decision.
- Conduct a cost/benefit analysis.

The only way to know the quality of a decision is through the test of time and results, so it is important to review the results once they are known and use them to improve your decision-making process. This is referred to as Continuous Process Improvement (CPI) and should be applied to all processes.

### **3.7. Step 5 - Solution Implementation**

Implementation of the selected solution requires the following steps:

- 1) Design
- 2) Build
- 3) Test
- 4) Deliver
- 5) Deploy
- 6) Support

Although the process is shown as a linear set of steps, in practice, it is an iterative process. Each step may yield information that affects earlier phases. A good design requires that new findings be used to improve earlier steps if appropriate.

### 3.8. Additional Resources

- ❖ Lumsdaine, E. Creative Problem Solving and Engineering. (1999) Prentice Hall.
- ❖ Eide, A. Engineering Fundamentals & Problem Solving. (2002) McGraw Hill

### 3.9. Problems

- 
1. What are the five steps of Creative idea generation in the correct order?

---

  2. Name the steps in the Creative problem solving that are Convergent.

---

  3. What are the four components of a completed Problem Definition?

---

  4. Use a Fishbone Diagram to perform a root cause analysis for a computer that does not turn on.

---

  5. What are the four rules of brainstorming?

---

  6. Describe the Anonymous method and the advantage of this method over Pin Card method.

---

  7. What is the most appropriate brainstorming process for a large team with 100 members?

---

  8. Describe the weighted advantage and disadvantage ranking technique?

---

  9. List at least 5 different decision-making methods in Optimal Solution Selection step.

---

  10. What are the top three problems that you are interested in and may have technology solutions? Explain the reasons for your choices.

---

  11. Identify a problem that can be addressed using computing/electrical related skills and technology. Apply the five-step Creative Problem-Solving technique to the selected problem.  
*Note: Be prepared to present in class.*

## **Chapter 4. Electrical Circuits**

### **4.1. Key Concepts and Overview**

- ❖ Charge, Current and Voltage
- ❖ Circuit Model
- ❖ Power Sources
- ❖ Kirchoff's Laws and DC Circuit Analysis
- ❖ Circuit Simplification

## 4.2. Charge, Current and Voltage

### Electric Charge

Charge in nature may be electric, color or magnetic. This section is focusing on electric charge. Electric charge is a basic property of electrons, a subatomic particle. Electrons by convention have a charge of -1 where protons have an opposite charge of +1. In general, electric charge is bipolar, meaning that electrical effects are described as having positive and negative charges.

The International Standard (SI) unit for electric charge is the Coulomb, which represents approximately  $6.24 \times 10^{18}$  elementary charges. Elementary charge is the charge on a single electron.

### Electric Current and Voltage

Electrical effects are attributed to both the separation of charge (voltage) and motion of charge (current).

- **Current**

Current can be thought of as the “speed or flow rate” of electrons. The current is measured in Amperes and at its simplest form is the change of charge  $q$  in one second. Mathematically current is expressed as:

$$i = \frac{\Delta q}{\Delta t} \quad \text{where}$$

$i$  = current in amperes  
 $q$  = charge in coulombs  
 $t$  = time in seconds

- **Voltage**

Voltage can be thought of as the “Potential difference” of electricity. Voltages are measured in volts between two points and represent the amount of energy required to move electric charge from point a to point b. Mathematically, current is expressed as:

$$v = \frac{\Delta e}{\Delta q} \quad \text{where}$$

$v$  = voltage in volts  
 $e$  = energy in joules  
 $q$  = charge in coulombs

Since we all have personal sensory experience with water flow and pressure, it may be helpful to compare water flow through a pipe with electricity through a wire.

WATER	ELECTRICITY
Pressure between two points	Voltage between two points
Flow Rate in pipe	Current in wire
Amount of water	Charge
Resistance of Pipe	Resistance of wire

Let's look at the water analogy. Suppose we have a faucet, a hose, and a 5-gallon bucket. If we connect the hose, put the other end in the bucket, turn on the faucet, and the bucket fills in one minute. We know the water is flowing through the hose at a rate of 5 gallons per minute. That rate of 5 gallons a minute is the same through the whole length of the hose, from the faucet until it empties into the bucket.

If you empty the bucket and put the hose back in the bucket, but this time we pinch the hose to slow the flow of water so now the bucket takes 2 minutes to fill, the flow rate is now 2.5 gallons per minute. The rate though the whole length of the hose is now 2.5 gallons per minute. The water pressure at the faucet is the same, but with increased resistance in the hose, the flow rate is decreased.



Now if we empty the bucket again, put the hose back in it, keep the hose pinched as it was before, but we open the faucet more so we have 5 gallons per minute flowing through the hose we will again fill the bucket in one minute. To some extent for a given resistance, the pinch of the hose, we can increase the pressure to achieve a lower flow rate, or we could decrease the resistance to achieve a higher flow rate. Of course, if we were to pinch the hose tight enough, we would completely stop the flow of water.

Electricity behaves in a similar way as water. In an electrical circuit if we increase the pressure, the voltage, and the resistance is unchanged then the current will increase. If we increase the resistance and the voltage stays the same, the current will decrease.

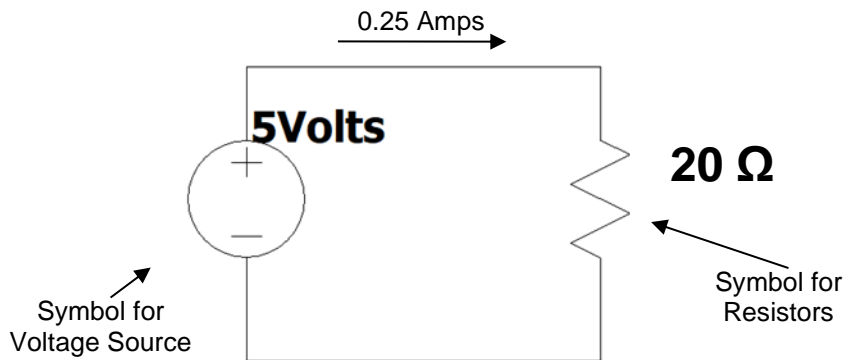
The units for voltage, current, and resistor are shown in the table below:

	Unit	Symbol
Voltage	Volts	V
Current	Amperes (amps)	A
Resistance	Ohms	$\Omega$ (the Greek letter omega)

Ohm's Law governs the relationship between voltage, current, and resistance. Electrical circuit analysis and design relies on use of Ohm's Law. The following three algebraic expression are equivalent, and all represent Ohm's law:

$$V = I * R \quad \text{or} \quad I = V / R \quad \text{or} \quad R = V / I$$

Here is an application of Ohms law to the following circuit:



The current and resistance is given in this example so we can use Ohm's Law ( $V=I*R$ ) to calculate the voltage.

$$0.25 \text{ amps} * 20 \text{ ohms} = 5 \text{ volts}$$

We could calculate the current by solving Ohm's Law for current  $I = \frac{V}{R}$ .

$$0.25 \text{ amps} = \frac{5 \text{ Volts}}{20 \text{ ohms}}$$

Finally, if we didn't have the resistance, we could calculate it by solving Ohm's Law for resistance  $R = \frac{V}{I}$ .

$$20 \text{ ohms} = \frac{5 \text{ Volts}}{0.25 \text{ amps}}$$

Normally when we are writing these equations, we would use the abbreviations V for volts, A for amps, and  $\Omega$  for ohms. So, we would normally write the equation above as:

$$V = 0.25 \times 20 = 5 \text{ V}$$

$$I = \frac{5}{20} = 0.25 \text{ A}$$

$$R = \frac{5}{0.25} = 20 \Omega$$

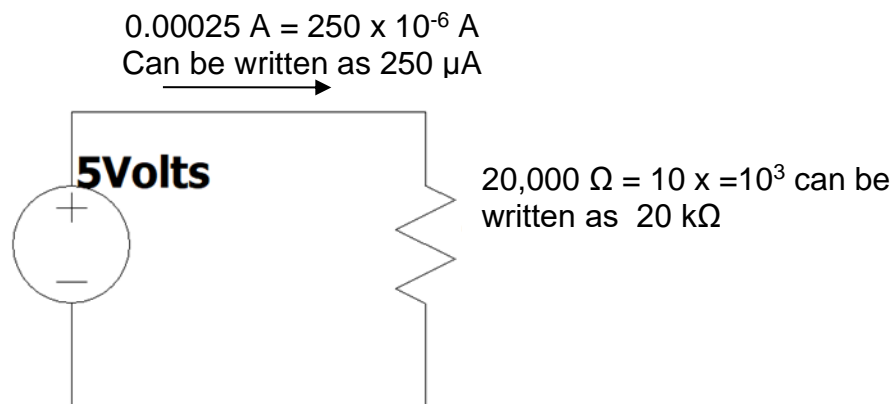
### Powers of 10

Engineering number notation use powers of 10 that are multiple of three in order to simplify communication of very small and very large numbers. Here are the most commonly used powers of 10 with the corresponding names:

Power of 10 larger than 1		Powers of 10 smaller than 1	
$10^{24}$	yotta (Y)	$10^{-3}$	milli (m)
$10^{21}$	zetta (Z)	$10^{-6}$	micro ( $\mu$ or u)
$10^{18}$	exa (E)	$10^{-9}$	nano (n)
$10^{15}$	peta (P)	$10^{-12}$	pico (p)
$10^{12}$	tera (T)	$10^{-15}$	femto (f)
$10^9$	giga (G)	$10^{-18}$	atto (a)
$10^6$	mega (M)	$10^{-21}$	zepto (z)
$10^3$	kilo (k)	$10^{-24}$	yocto (y)

*Note: Small and capital forms of the same letter refer to different powers of 10.*

Below is an example of ohm's law that shows how using names of Power of 10 helps simplify values for a given circuit:



### 4.3. Ideal Circuit Model

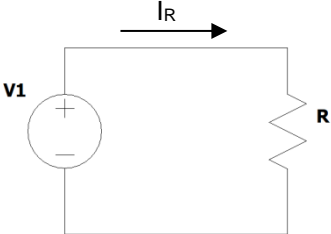
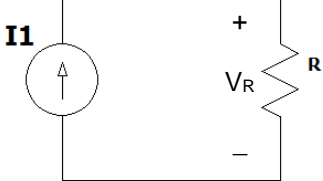
An electrical circuit is a mathematical model that approximates the behavior of an actual electrical system. We will be using Ideal Circuit Theory which makes the following three assumptions:

- 1) Electricity moves instantly through an ideal circuit
- 2) Current through all components in a branch (wire) is the same
- 3) Total power in a valid ideal circuit is zero

The first step in circuit analysis is learning about the components and rules that govern their operations. We will be using four basic elements to model electrical systems. In this chapter, only power source and resistors are discussed.

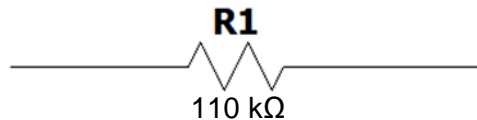
#### Power Sources (or Power Supplies)

There are two basic independent types of ideal power sources: voltage source and current source. The following table compares the two:

Voltage Source (Ideal, Independent)	Current Source (Ideal, Independent)
<ul style="list-style-type: none"> <li>• Supplies a constant voltage regardless of the rest of the circuit.</li> <li>• Does not consume any of the power internally.</li> </ul>  <p>'V<sub>1</sub>' is a reference designator which is used to identify this particular voltage source in the circuit. If V<sub>1</sub> = 5 V, voltage source keeps the voltage at 5 Volts, but varies the current as value of R changes to satisfy the Ohm's Law (<math>I_R = 5/R</math> A).</p>	<ul style="list-style-type: none"> <li>• Supplies the designated current regardless of the rest of the circuit.</li> <li>• Does not consume any of the power internally.</li> </ul>  <p>'I<sub>1</sub>' is a reference designator which is used to identify this particular current source in the circuit. If I<sub>1</sub> = 2 A, current source keeps the current at 2 Amps, but varies the voltage as value of R changes to satisfy the Ohm's Law (<math>V_R = 2 * R</math> V).</p>

#### Resistor

Resistors provide resistance which is the ability to resist or slow the flow of current. In ideal circuits, the assumption is that wires don't have resistance ( $R=0$ ). Only resistors have resistance. Resistance is measured in ohms ( $\Omega$ ). Below is the resistor symbol.



Note:

- \* R1 is the reference designator
- \* Resistance is 110 k $\Omega$  or 110,000 $\Omega$ .

#### 4.4. Power Calculation

Power is the measurement of ability of electrical circuit to do work such as heating, moving or other functions. The most general equation for calculating Power is shown below:

$$P = V * I \quad \text{where:}$$

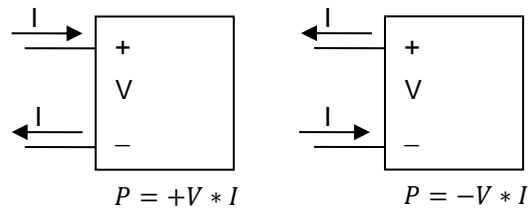
$P = \text{power in Watts}$   
 $I = \text{Current in Amps}$   
 $V = \text{Voltage in Volts}$

For resistors, power may also be written in the forms by applying Ohm's law:

$$P = \frac{V^2}{R} \quad \text{or} \quad P = R * I^2$$

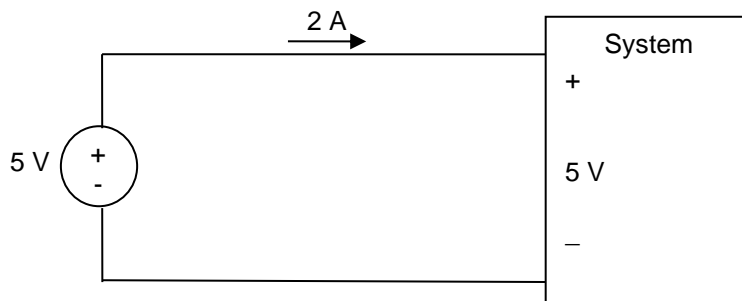
An electrical element such as resistors consumes power. Elements that consume power are referred to as passive elements and value of power is always positive. On the other hand, an electrical element such as voltage source that generates power is referred to as active element and the value of power is always negative.

These definitions are part of Passive Convention which requires the current enter the positive terminal of element when calculating the power: If current is entering the negative terminal, then you need to add a negative sign to your power calculation as shown below:



#### Example A – Power Calculation

Find the Power at the voltage source and at the system:



#### Solution

For the box labeled system, the current is entering the positive terminal so no need to add a negative in the power calculation:

$$P_{\text{system}} = V * I = 5 * 2 = 10W$$

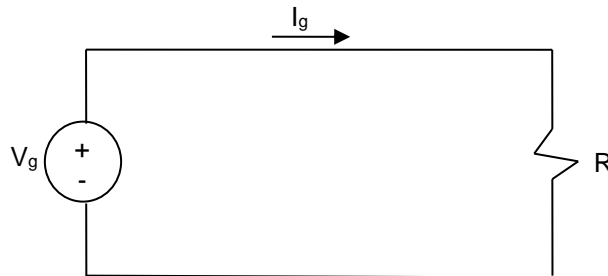
$P > 0$  therefore the voltage source is a passive device consuming power

For the voltage source, the current is leaving the positive terminal so need to add a negative sign to the power calculation:

$$P_{\text{source}} = -V * I = -(5 * 2) = -10\text{W} \text{ (Voltage source is an active device } P < 0 \text{ therefore the voltage source is an active device generating power.)}$$

**Example B – Power Calculation**

Find the value of R and the power consumed by the resistor if  $V_g = 1 \text{ kV}$  and  $I_g = 5 \text{ mA}$ .



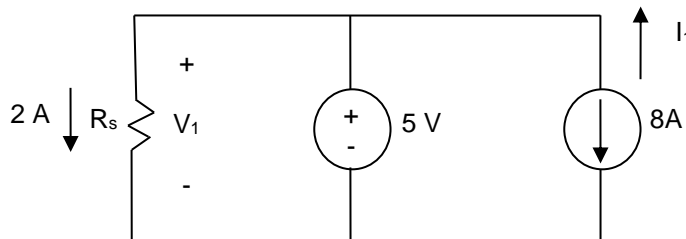
**Solution**

$$R = \frac{V_g}{I_g} = \frac{1000}{0.005} = 200,000 \Omega = 200 \text{ k}\Omega$$

$$P_r = I_g^2 \times R = (0.005)^2 \times (200,000) = 5 \text{ W}$$

**Example D – Ideal Circuit Properties**

Determine the current  $I_1$ ,  $V_1$  and  $R_s$  for the following valid circuit:



**Solution**

Current is the same throughout a branch or wire:

$$I_1 = -8\text{A}$$

Voltages are same when both ends of elements are connected:

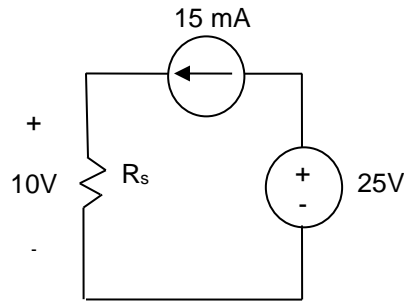
$$V_1 = 5\text{V}$$

Apply Ohm's Law:

$$R_s = \frac{V}{I} = \frac{5\text{V}}{2\text{A}} = 2.5\Omega$$

**Example E – Valid Circuit (Total power = 0)**

Calculate the value of  $R_s$  and determine if the following the circuit is a valid ideal circuit:



Hint: A circuit is valid if total power is 0 or the magnitude of power generated is equal to power consumed.

**Solution**

In an ideal circuit current, at any part of a branch is the same.

$$R_s = \frac{V}{I} = \frac{10V}{0.015A} = 667\Omega$$

In a valid circuit, the sum of power is 0.

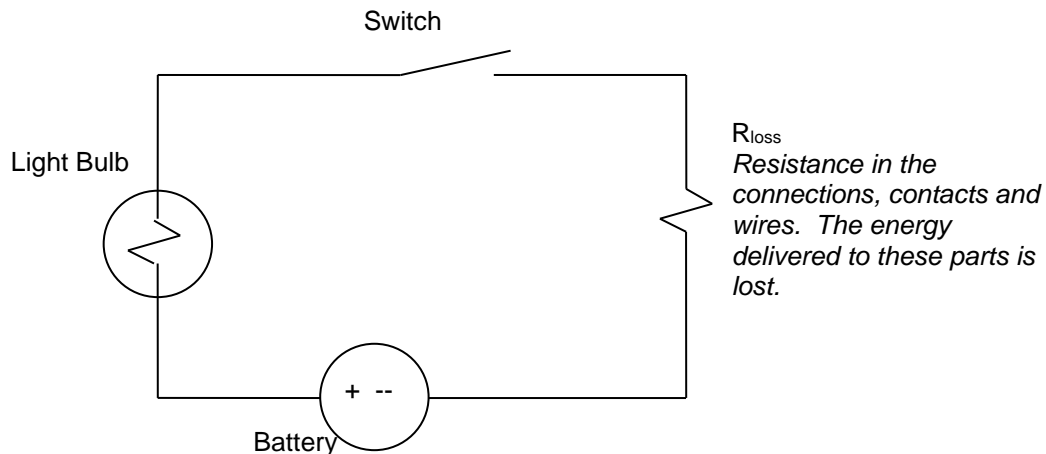
$$\begin{aligned} \Sigma \text{ power} &= P_{\text{resistor}} + P_{\text{voltage source}} + P_{\text{current source}} \\ &= (10V \times 0.015A) - (25V \times 0.015A) + ((25V - 10V) \times 0.015A) \\ &= 0W \end{aligned}$$

Therefore, this circuit is valid.

**Example F – Modeling**

Construct a circuit model for a flash light using electrical components.

**Solution**



**Light bulb** is a resistor that converts electrical energy to light in visible range.

**Battery** is the power source. It is not an ideal source since the voltage drops over the life of the battery.

**Switch** to connect and disconnect power to turn light bulb off or on.

Other considerations include Heat generation, Power requirements.

**Example G – Modeling**

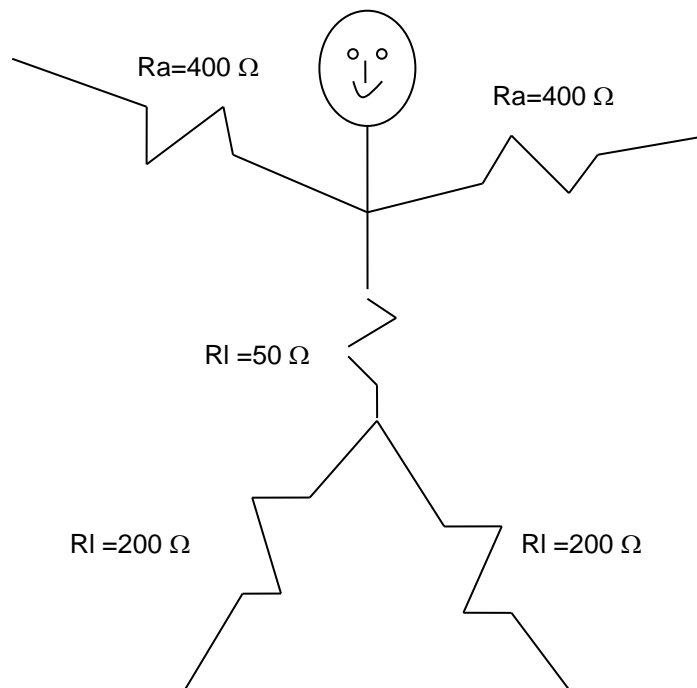
Construct an electrical circuit model for the human body.

**Solution**

For an average human, arm resistance is  $400 \Omega$ , torso resistance is  $50 \Omega$  and leg resistance is  $200 \Omega$ . As you can see the ticker parts have lower resistance. This model is used for electricity safety training.

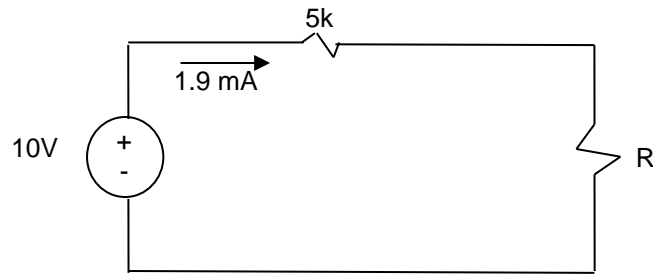
High voltage is rarely the cause of death although it causes burns which could be serious. On the other hand, the effect of electrical current on human nervous system is very severe, especially if the nerves are the ones controlling the heart muscles. The following is approximation of current levels and the corresponding physiological responses:

Current	Physiological Responses
3-5 mA	Barely Perceptible
35-50 mA	Extreme Pain
50 – 70 mA	Muscle Paralysis
> 500 mA	Heart Stoppage



**Student Exercise A – Power Calculation**

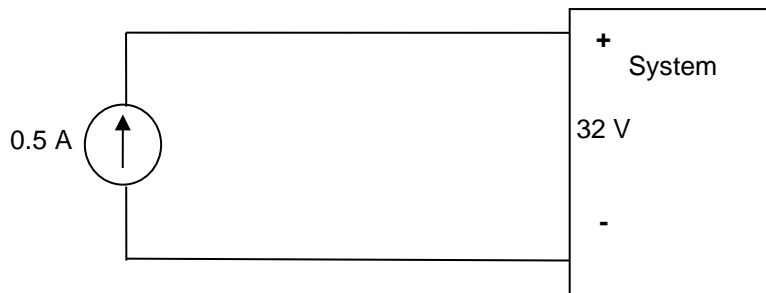
Find the value of R in the following circuit and determine if the following circuit is a valid ideal circuit:



**Solution**

**Student Exercise B – Power Calculation**

Find the Power at the current source and Power at the system for the following circuit. Also, specify if the power is being consumed or generated.:



**Solution**



**Student Exercise C – Modeling**

What is the physiological impact of 220 V being shorted between the person's left hand and right leg?

**Solution**

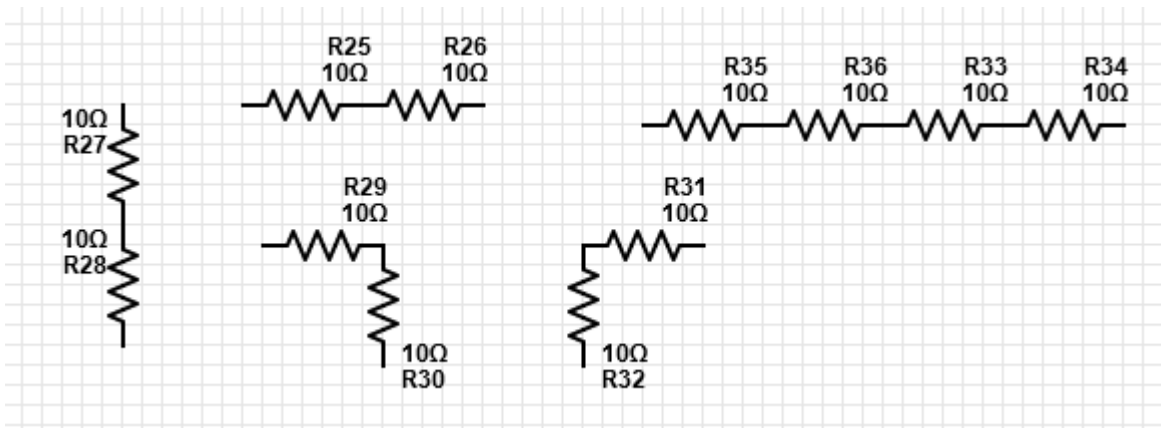
## 4.5. Resistor Simplifications

So far, the examples have only shown a single voltage supply and a single resistor. We can calculate the current in a circuit with a single voltage supply and a single resistor by simply dividing the voltage by the resistance (Ohm's Law). This section shows how to reduce multiple resistors to a single resistor so we can calculate the current. We will then show how to use that current to figure out the current and voltage drop for each resistor in the circuit.

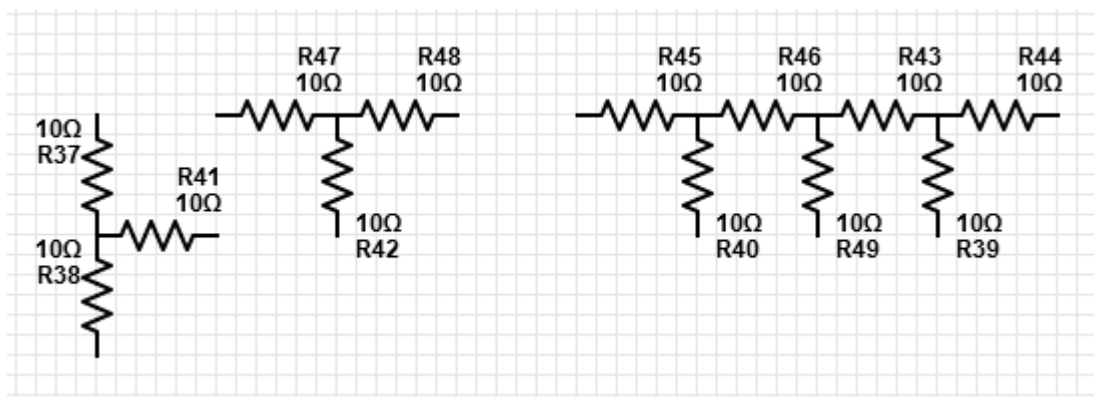
For a significant number of circuits, analysis may only require replacing multiple resistors with a single equivalent resistor. This section discusses two types of resistor networks that may be replaced by an equivalent resistor: series resistor network and parallel resistor network.

### Resistors in Series

For two resistors to be in series, they must be connected at a single end and have the same current. Here are examples of resistors in series:

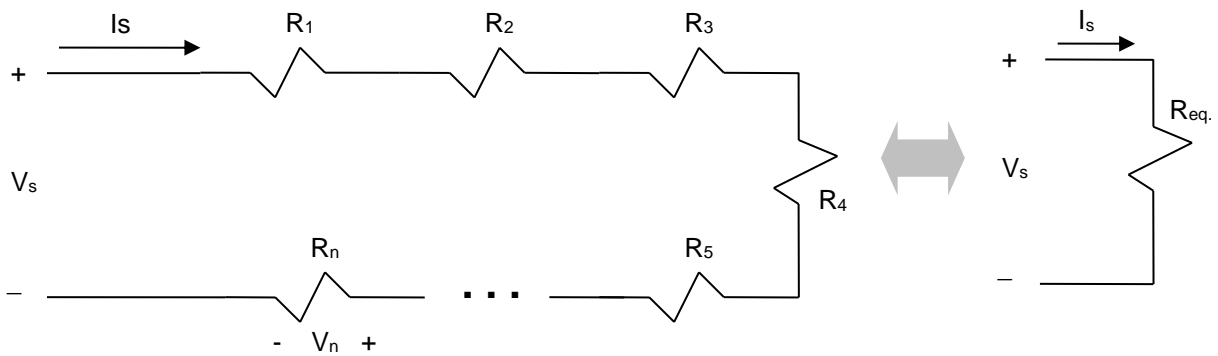


Note that the current through series resistors are the same. Now, if we allow the current to divide then the resistors are not in series. Here are examples of resistors which are not in series:



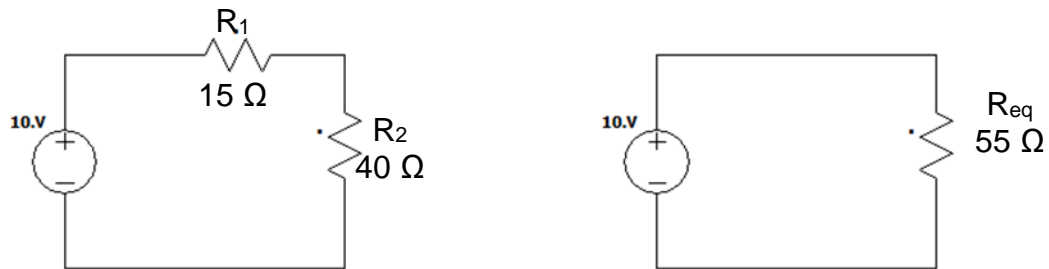
If your circuit has two or more resistors in series, then you can use the following generalized equation to find the equivalent resistor and replace them:

$$R_{eq} = R_1 + R_2 + R_3 + \dots + R_n = \sum_{i=1}^n R_i \text{ for series resistors}$$



**Example A – Series Resistor Simplification**

Calculate the current from Voltage Source in the following circuit.

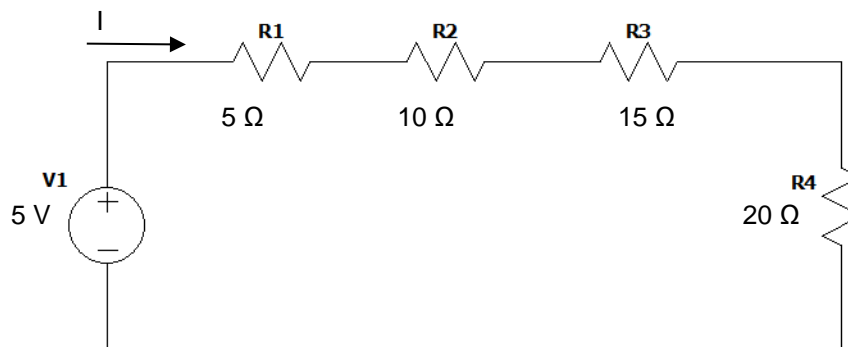


$$R_{eq} = R_1 + R_2 = 40 + 15 = 55 \Omega$$

The equivalent resistance is  $55\Omega$ . Apply Ohm's law to find the current is  $I = \frac{10V}{55\Omega} = 0.18A$

**Example B –Series Resistor Simplification**

Calculate the current from Voltage Source for the following circuit:

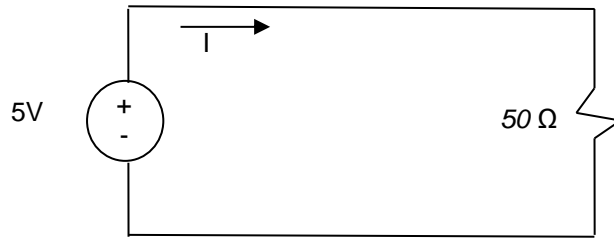


**Solution:**

Step1) Find the equivalent of the four series resistors:

$$R_{eq} = R_1 + R_2 + R_3 + R_4 = 5 + 10 + 15 + 20 = 50 \Omega$$

Resulting in the following equivalent circuit:



Step2) Apply the Ohm's Law and find current  $I = V/R = 5/50 = 0.1 \text{ mA}$

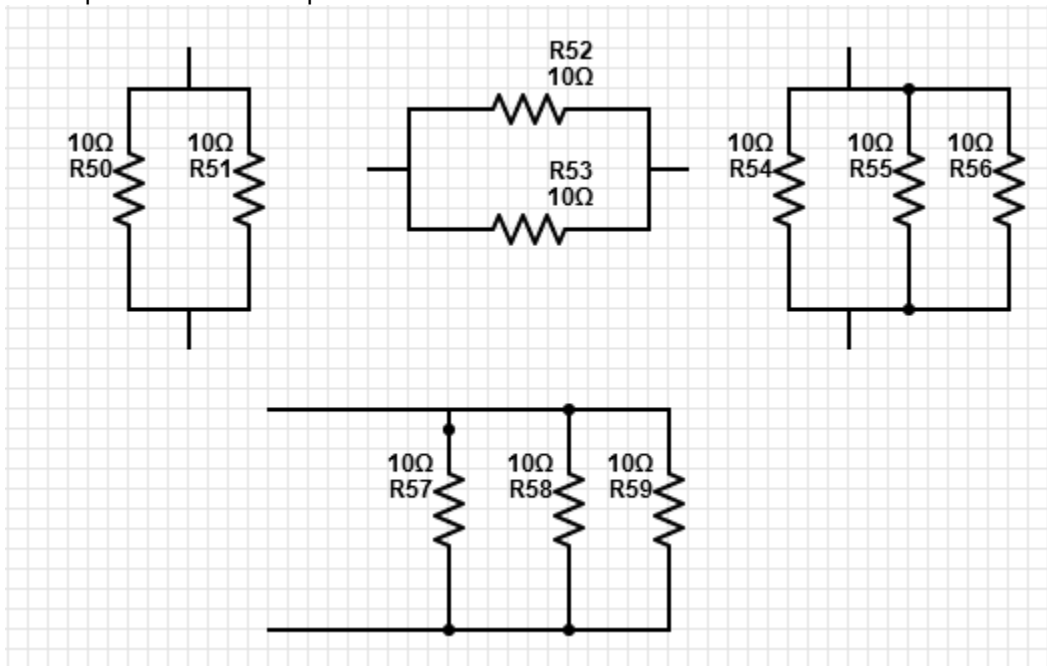
Step3) Using the Current  $I$ , we can go back to the original circuit and calculate the voltages in the 4 original resistors:

$$\begin{aligned} V_{R1} &= 0.1 * 5 \Omega = 0.5 \text{ V} \\ V_{R2} &= 0.1 * 10 \Omega = 1.0 \text{ V} \\ V_{R3} &= 0.1 * 15 \Omega = 1.5 \text{ V} \\ V_{R4} &= 0.1 * 20 \Omega = 2.0 \text{ V} \end{aligned}$$

Totalling the four voltages together, we get 5V which is the supply voltage. A good way to check your work.

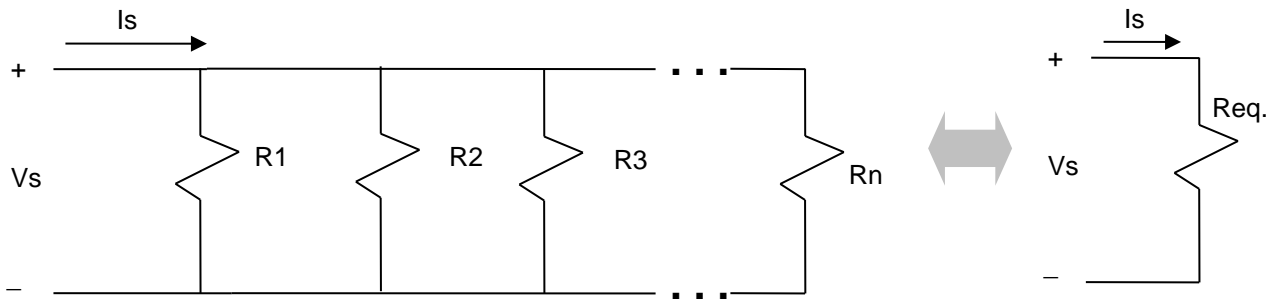
### Resistors in Parallel

For two resistors to be in Parallel, they must be connected at both ends and have the same voltage. Here are examples of resistors in parallel:



The equivalent resistor for a parallel resistor network can be calculated using the following generalized equation:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots + \frac{1}{R_n} = \sum_{i=1}^n \frac{1}{R_i}$$



**Example A – Parallel Resistor Simplification**

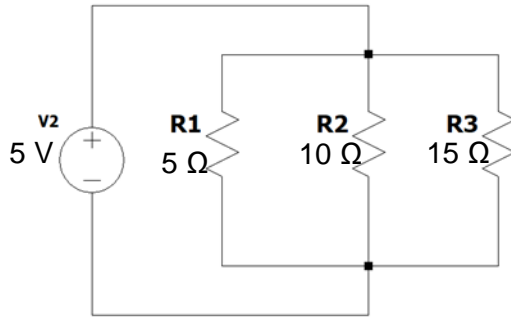


$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} \rightarrow R_{eq} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}} = \frac{1}{\frac{1}{10} + \frac{1}{40}} = 8 \Omega$$

The current through equivalent  $8\Omega$  resistor is  $\frac{10V}{8\Omega} = 1.25A$ . The voltage drop across each resistor in the original circuit is the same,  $10V$ . Therefore, we have current through  $10\Omega$  resistor ( $\frac{10V}{10\Omega} = 1A$ ) plus the current through  $40\Omega$  resistor ( $\frac{10V}{40\Omega} = 0.25A$ ) for a total of  $1.25A$  same as the equivalent circuit.

**Example B – Parallel Resistor Simplification**

Calculate the equivalent resistance for the following circuit:



**Solution**

Resistor R1, R2 and R3 are in parallel so we can use the following equation to find the equivalent resistance:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} = \frac{1}{5} + \frac{1}{10} + \frac{1}{15} \rightarrow R_{eq} = 2.73 \Omega$$

Although the example did not ask for it, we can find the current in each of the resistors since they all resistors have the same voltage of 5V across them:

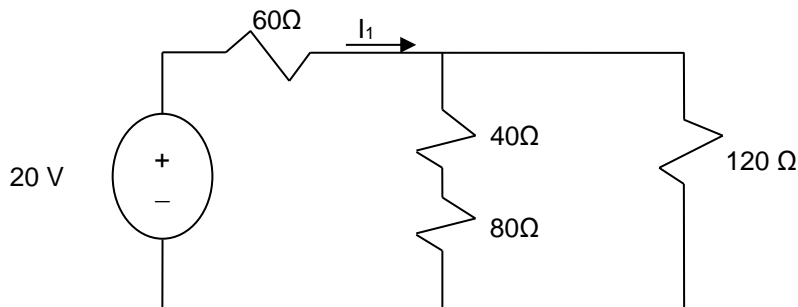
The current through R4 is  $\frac{5}{5} = 1.0A$ .

The current through R6 is  $\frac{5}{10} = 0.5A$ .

The current through R5 is  $\frac{5}{15} = 0.333A$ .

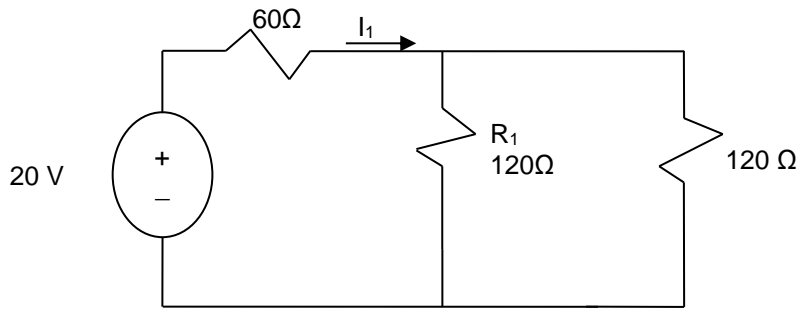
**Example C – Circuit with series and parallel resistors**

For the following circuit determine the value of  $I_1$  using the resistor simplifications.



**Solution**

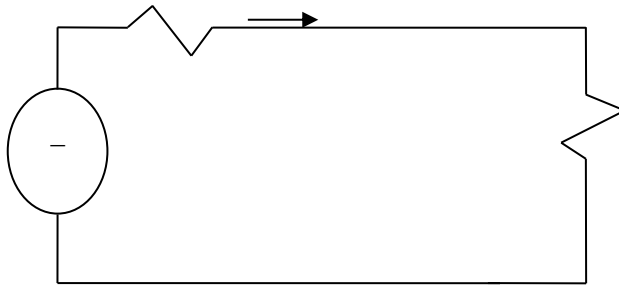
First, we see that the 40Ω and 80Ω are in **series**. Therefore, they can be replaced with  $R_1 = 40\Omega + 80\Omega = 120\Omega$ . then the new circuit is shown below:



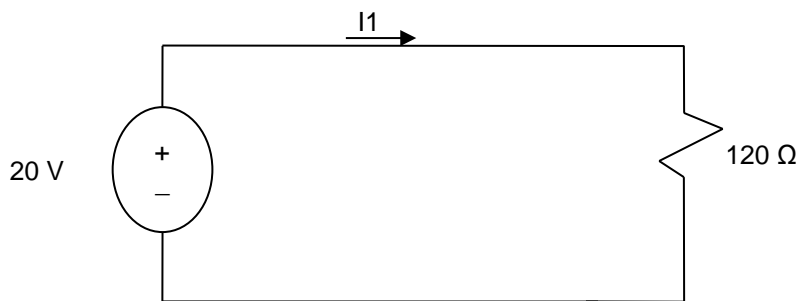
No we see that the new  $R_1 = 120\Omega$  and original  $120\Omega$  are in parallel. Therefore, we can find the equivalent  $R_2$  by using the equivalent for parallel resistors equation:

$$R_{eq2} = \frac{1}{\frac{1}{120\Omega} + \frac{1}{120\Omega}} = 60\Omega$$

Therefore, the circuit can be redrawn as:

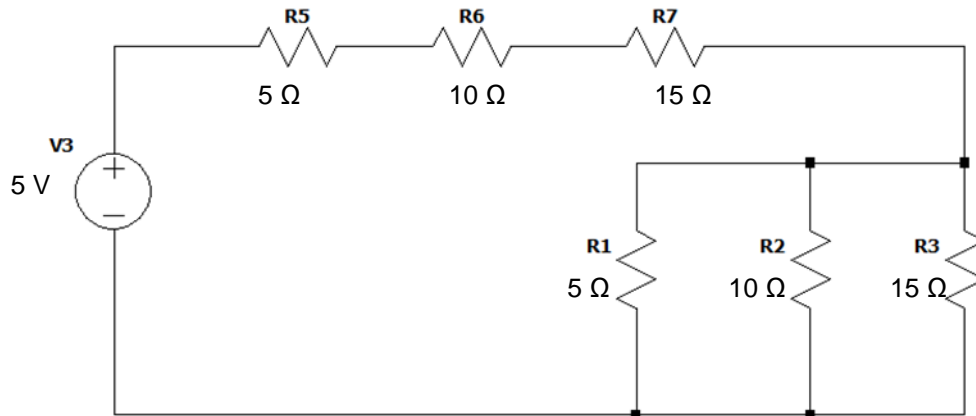


Finally, the two  $60\Omega$  resistors are in series, so they can be replaced with  $R_3 = 60 + 60 = 120\Omega$



$$I_1 = \frac{V}{R} = \frac{20}{120} = \frac{1}{6} A$$

**Example D** – Circuit with series and parallel resistors  
Use the following circuit to answer the questions in order:

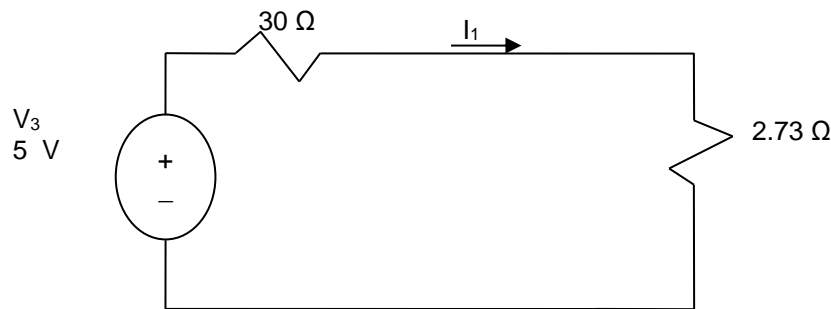


**Figure 1**

1. What is the equivalent resistance for the above circuit?

For the first set of resistors  $R_1$ ,  $R_2$ ,  $R_3$  which are in parallel, we calculate  $R_{eq}=2.73 \Omega$ .  
 For the second set of resistors  $R_5$ ,  $R_6$ ,  $R_7$  which are in series, we calculate  $R_{eq} = 30 \Omega$ .

Now we can reduce the circuit to:



**Figure 2**

The equivalent resistance is  $30\Omega + 2.73\Omega = 32.73\Omega$ .

2. What is the current through that equivalent resistor?

The current through the equivalent resistor is  $\frac{5V}{32.73\Omega} = 153mA$ .

3. What is the current through each resistor?

Now is time to figure out the current through each resistor. The current through the equivalent resistor is  $153mA$ . Since that resistor is equivalent to the  $30\Omega$  and  $2.73\Omega$  resistors in Figure 2, and those resistors are in series, then the current through both those resistors is  $153mA$ .



The  $30\Omega$  resistor represents  $R5$ ,  $R6$ , &  $R7$  in Figure 1, and those resistors are in series, so they also have a current of  $153\text{mA}$ .

The current through the  $2.73\Omega$  resistor, which is equivalent to parallel resistors  $R1$ ,  $R2$ , and  $R3$  is  $153\text{mA}$  which is the sum of the currents through  $R1$ ,  $R2$ , and  $R3$ .

In order to find the current through each of  $R1$ ,  $R2$ , and  $R3$ , first need to find the voltage drop across the  $2.73\Omega$  resistor. Earlier we realized that the current through the  $2.73\Omega$  resistor was  $153\text{mA}$ , so the voltage drop is  $2.73\Omega * 153\text{mA}$  or  $418\text{mV}$ .

Knowing the voltage drop, we can now calculate the current through each resistor.

The current through  $R1$  is  $\frac{418\text{mV}}{5\Omega} = 83.6\text{mA}$ .

The current through  $R2$  is  $\frac{418\text{mV}}{10\Omega} = 41.8\text{mA}$ .

The current through  $R3$  is  $\frac{418\text{mV}}{15\Omega} = 27.9\text{mA}$ .

When we add up these three currents it comes to  $153.3\text{mA}$  which is very close to the  $153\text{mA}$  we expected.

4. What is the voltage drop across each resistor?

We just calculated the voltage drop across  $R10$ ,  $R11$ , and  $R12$  as  $418\text{mV}$ .

The voltage drop across  $R5$  is  $153\text{mA} * 5\Omega = 765\text{mV}$ .

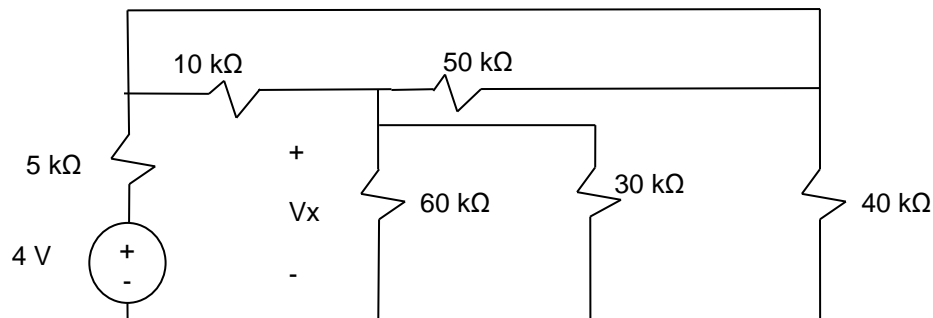
The voltage drop across  $R6$  is  $153\text{mA} * 10\Omega = 1.53\text{V}$ .

The voltage drop across  $R7$  is  $153\text{mA} * 15\Omega = 2.3\text{V}$ .

If we add up these three voltages, it comes to  $4.595\text{V}$ , and if we also add in the voltage drop across the parallel resistors it comes to  $5.013\text{V}$  which is very close to our source voltage of  $5\text{V}$ . The difference of  $0.013$  volts or  $3\%$  is due to calculation precision error (not carrying enough digits after the decimal point)

### Student Exercise A – Resistor Simplification

Use Resistor simplification to find the current flow through  $4\text{V}$  sources and the power generated by the  $4\text{V}$  source.



### Solution



## 4.6. Common Terms Used in Circuit Analysis & Examples

### Usage of Term Voltage

In referring to voltage, we might use terms like:

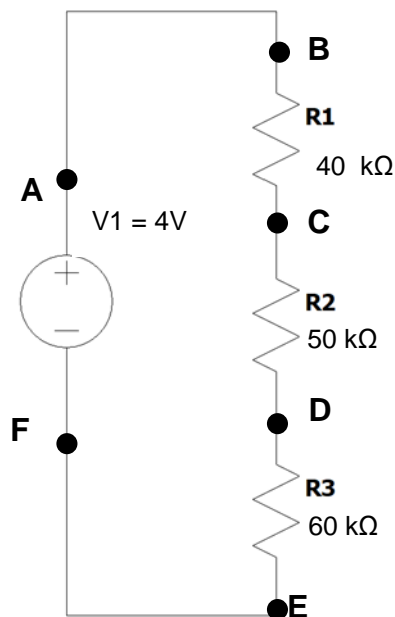
- What is the voltage at  $V_1$ ?
- What is the voltage at some reference point?
- What is the voltage drop across  $R_{23}$ ?

Whenever we make a voltage measurement, we are always making it relative to another point in the circuit.

If we ask, “What is the voltage at  $V_1$ ?”, we mean what is the voltage relative to  $V_{ref}$  or the negative side of the power supply.

If we ask, “What is the voltage drop across  $R_{23}$ ?” we are asking for the difference in voltage between one side of a resistor and the other.

In the circuit below we have labeled some points A through F. We will use these points to give some examples of how we might phrase statements about voltage.



1. What is the voltage at A?  $4V$  We assume that it is relative to the negative terminal of the voltage supply which is F.
2. What is the voltage at B? Still  $4V$ . There is no resistor between A and B so the voltage is the same, no voltage drop or  $V_{BC} = 0$ .
3. What is the voltage drop across  $R_1$ ? Here we would measure the difference between the voltage at B and the voltage at C.
4. What is the voltage at C? This time it would be the voltage a C relative to F.
5. What is the voltage drop across  $R_2$ ? Here we would measure the difference between the voltage at C and the voltage at D.

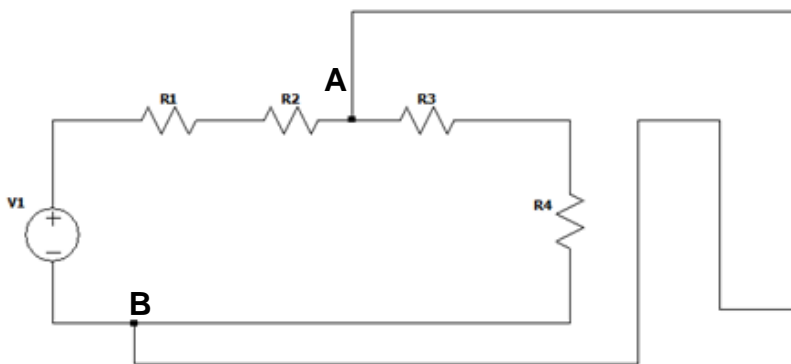
6. If we asked, "What is the voltage at D?", or "What is the voltage drop across  $R_3$ ?" The answers would be the same.

### Opens and Shorts

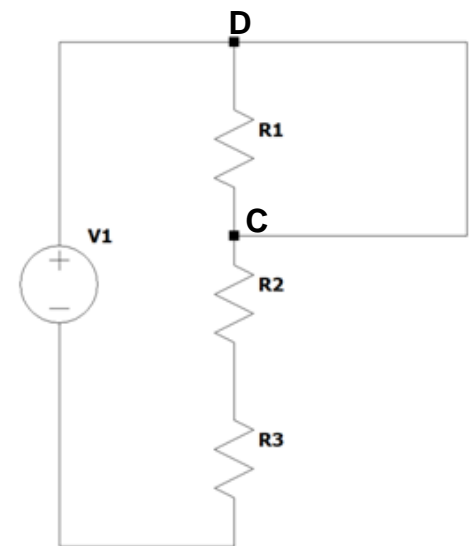
Typically, engineers do not design an open or a short into their circuits. Opens and Shorts are manufacturing defects, design errors or configuration settings.

#### Short ( $R = 0$ )

Short is a connection which allows the current to bypass a circuit element. It's acceptable to think of it as a shortcut but it is not always a shorter physical path. Short is a path of no resistance ( $R=0$ ). If we can trace a path from one end of an element to the other end of the same element without passing through another element, that is a short. Here are two examples of short:



In this circuit  $R_3$  and  $R_4$  are shorted so no current flows through them. The equivalent voltage and resistance between points A and B is 0.

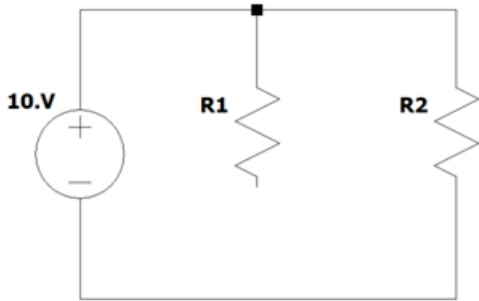


In this circuit  $R_1$  is shorted so no current flows through it. The equivalent voltage and resistance between points C and D is 0.

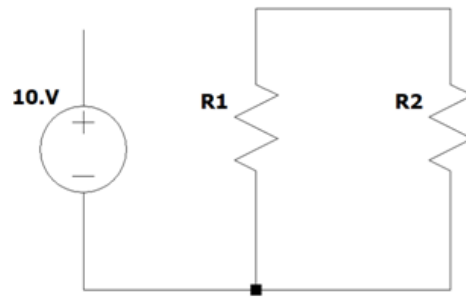
#### Open ( $R = \infty$ )

A missing connection is called an Open (Resistance is Infinite). Open is sometime called Open Loop. Most Ohmmeters display "OL" for Open Loop when they are measuring resistance more the upper limit of Ohmmeter.

For elements to receive current in a circuit, they need to have a path from the source to the element, through the element, and back to the other terminal of the source. There may be one or more other elements in the path. Here are a couple of Open examples:



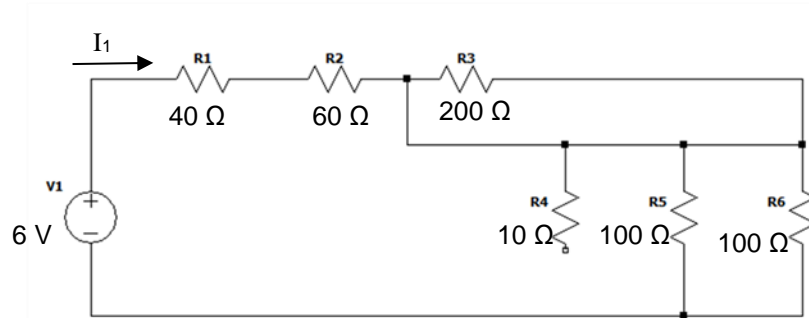
In this circuit, there is an open below R1 therefore the resistance is infinity in this branch. R1 branch has no current through it. But R2 branch has current equal to  $10/R_2$  through it.



In this circuit, there is an open above voltage source so no current will flow out. Neither R1 nor R2 received any current. Therefore R1 & R2 have no current through them.

### Example A – Analysis using Resistor Simplification

Determine the current through resistor R<sub>1</sub> and voltage across resistor R<sub>6</sub>.

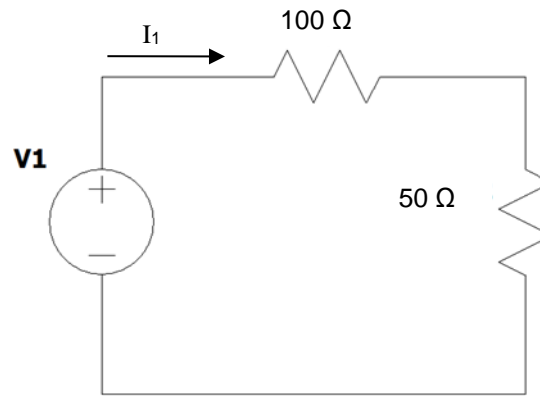


### Solution

Step 1) Find the equivalent resistance.

- \* R1 and R2 are in series so equivalent is  $100\ \Omega$
- \* R3 is in parallel with short ( $R=0$ ) so equivalent is  $0\ \Omega$
- \* R4 is open ( $R=\infty$ ) and in parallel with R5 and R4 so equivalent is  $50\ \Omega$

At this point we can redraw the circuit as:



Finally,  $100\ \Omega$  and  $50\ \Omega$  are in series for an equivalent of  $150\ \Omega$ .

Step 2) Apply Ohm's Law

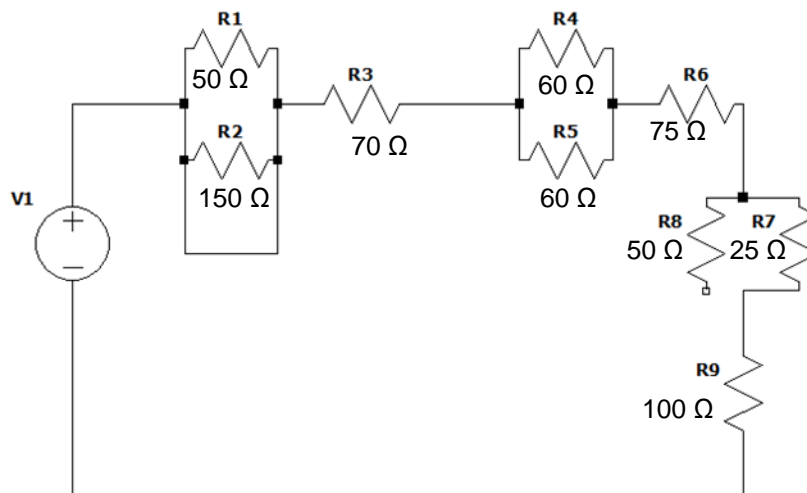
$$I_1 = V_1/R = 6/150 = 0.04\ \text{A} = 40\ \text{mA} \text{ which is the current through } R_1$$

Step 3) Find Voltage across  $50\ \Omega$  which is the same voltage across  $R_5$  and  $R_6$

$$V_{R6} = 0.04 * 50 = 2\ \text{V}$$

### Student Exercise A – Analysis using Resistor Simplification

Find the current through  $R_3$  and Voltage across  $R_7$  in the following circuit:



### Solution

## 4.7. Kirchhoff's Current Laws (KCL)

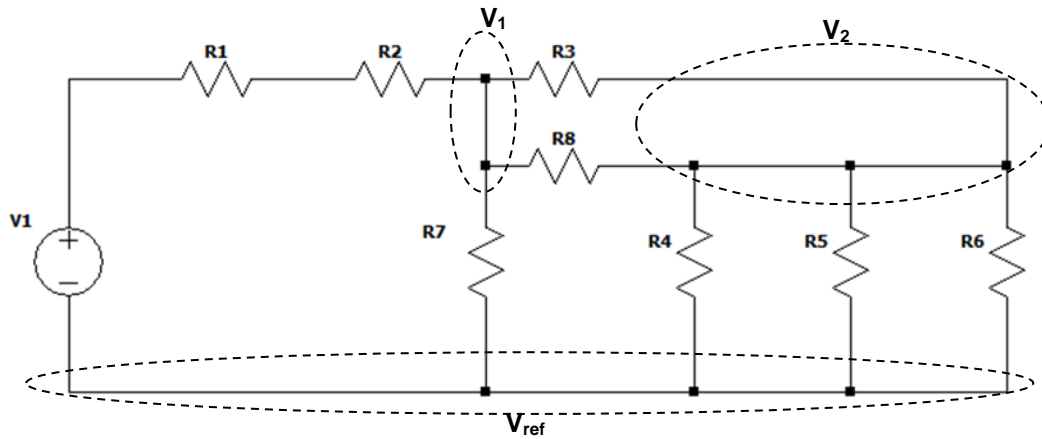
The circuits we have looked at so far only have one voltage source and are fairly simple. For more complicated circuits we will need to use a more sophisticated technique. Kirchhoff's Laws provides us with a powerful technique to analyze more complex circuits.

Kirchhoff's Laws have two equally powerful variations:

- Kirchhoff's Current Law (KCL)  
Sum of currents exiting any given node of an ideal circuit is equal to zero.
- Kirchhoff's Voltage Law (KVL)  
Sum of voltages around any given loop of an ideal circuit is equal to zero.

This section introduces Kirchhoff's Current Law (KCL). But first we need to define nodes and essential nodes. A node is where two or more elements are connected together. An essential node is a special node where more than two elements are connected together. Essential node is the only type node we use in KCL analysis so when discussing node in context of KCL, we always mean essential node.

Here is a sample circuit with 3 nodes (meaning essential nodes):



One node must always be designated as reference node ( $V_{ref}$ ) with voltage value of zero. Other nodes can be given voltage designations  $V_1$  and  $V_2$ . These node designations ( $V_1$  and  $V_2$ ) are referring to voltage from the node to  $V_{ref}$ .

Now that we have defined the node, here are the four steps to use KCL in circuit analysis:

- 1) Identify the essential nodes  
Pick one node as reference ( $V_{ref} = 0V$ ) and Label the rest as  $V_1, V_2, \dots$
- 2) Draw current arrows  
Current arrows are always drawn from the node out (positive direction is away from the node) and label currents  $I_1, I_2, \dots$
- 3) Write KCL equation for each node except  $V_{ref}$

The sum of currents at each node equals zero.  $\rightarrow I_1 + I_2 + I_3 + \dots = 0$

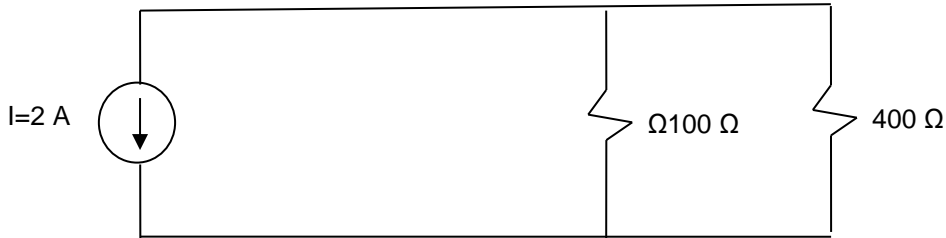
If the circuit has  $n$  essential nodes, write  $(n-1)$  KCL equation. You do not need to write the

equation for reference node  $V_{ref}$ .

- 4) Apply Ohm's Law ( $I = \frac{V}{R}$ ) to re-write the KCL equations in term of node voltages and given values of current.

**Example A - Kirchhoff's Current Law (KCL) Application**

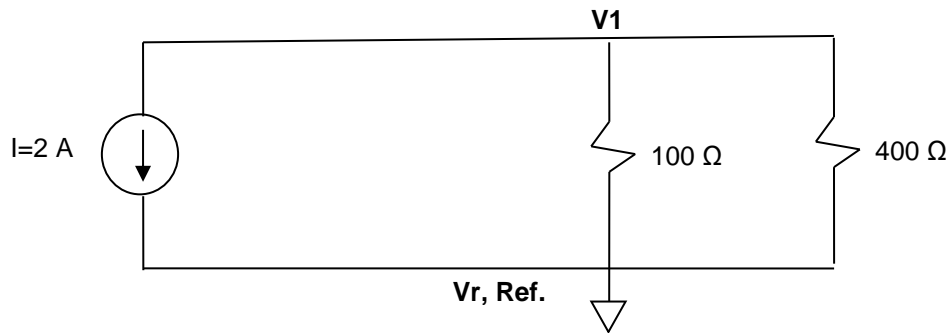
Use Kirchhoff's Current Law to find the voltage across  $400 \Omega$  resistor in the following circuit:



**Solution:**

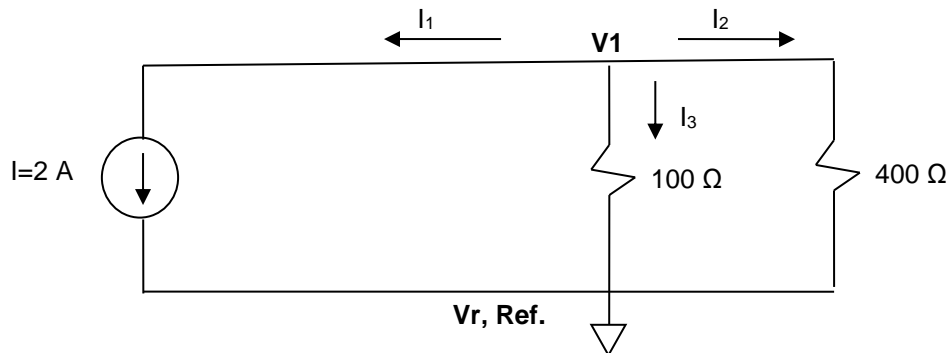
- 1) Identify the essential nodes

There are only two nodes, pick one node as reference ( $V_{ref} = 0$ ) and label the other node as  $V_1$ .



- 2) Draw current arrows

Draw all current arrows outward from the nodes (Positive) and label currents:  $I_1, I_2, I_3$ .





3) Apply KCL

We only have two essential nodes in this case so write (2-1) equation (only node  $V_1$ .)

$$\text{KCL @ } V_1 \rightarrow I_1 + I_2 + I_3 = 0\text{A}$$

4) Apply Ohm's Law ( $I = \frac{V}{R}$ ) to re-write the KCL equation(s) in term of Node voltage ( $V_1$ ) and known currents.

$$I_1 = 2\text{A}$$

$$I_2 = \frac{V_1 - V_{ref}}{400} = \frac{V_1 - 0}{400} = \frac{V_1}{400}$$

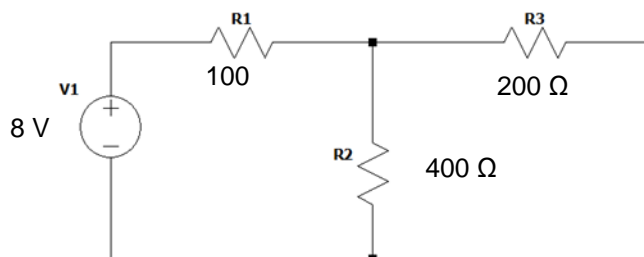
$$I_3 = \frac{V_1 - V_{ref}}{100} = \frac{V_1 - 0}{100} = \frac{V_1}{100}$$

Replace currents in KCL equation

$$\text{KCL @ } V_1 \rightarrow 2 + \frac{V_1}{100} + \frac{V_1}{400} = 0 \rightarrow V_1 = -160\text{ V}$$

### Example B - Kirchhoff's Current Law (KCL) Application

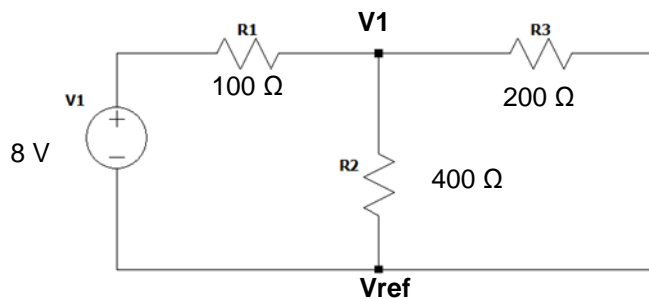
Find Voltage across the R3 using KCL.



### Solution

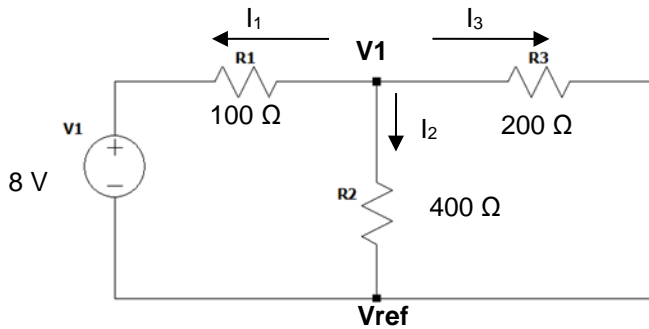
1) Identify the essential nodes

There are only two nodes, pick one node as reference ( $V_{ref} = 0$ ) and label the other node as  $V_1$ .



2) Draw current arrows

Draw all current arrows outward from the nodes (Positive) and label currents:  $I_1, I_2, I_3$ .



3) Apply KCL

We only have two nodes so write (2-1) equation (only node  $V_1$ .)

$$\text{KCL @ } V_1 \rightarrow I_1 + I_2 + I_3 = 0\text{A}$$

4) Apply Ohm's Law ( $I = \frac{V}{R}$ ) to re-write the KCL equation(s) in terms of Node voltage ( $V_1$ )

Determine value of currents in terms of Node Voltage ( $V_1$ ) & known currents.

$$I_1 = \frac{V_1 - 8}{100}$$

$$I_2 = \frac{V_1 - 0}{400} = \frac{V_1}{400}$$

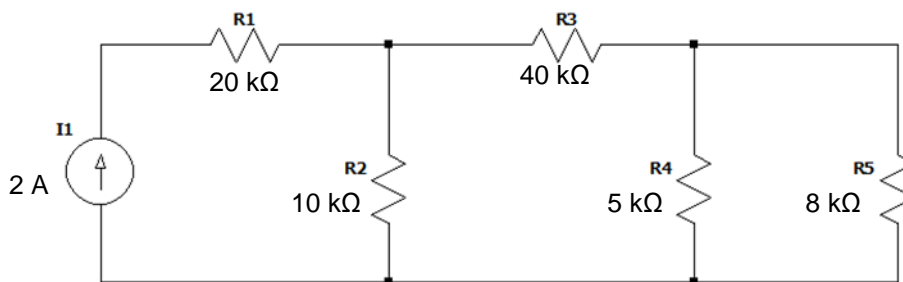
$$I_3 = \frac{V_1 - 0}{200} = \frac{V_1}{200}$$

Replace currents in KCL equation

$$\text{KCL @ } V_1 \rightarrow \frac{V_1 - 8}{100} + \frac{V_1}{400} + \frac{V_1}{200} = 0 \rightarrow V_1 = 4.57\text{ V "voltage across } R_3\text{"}$$

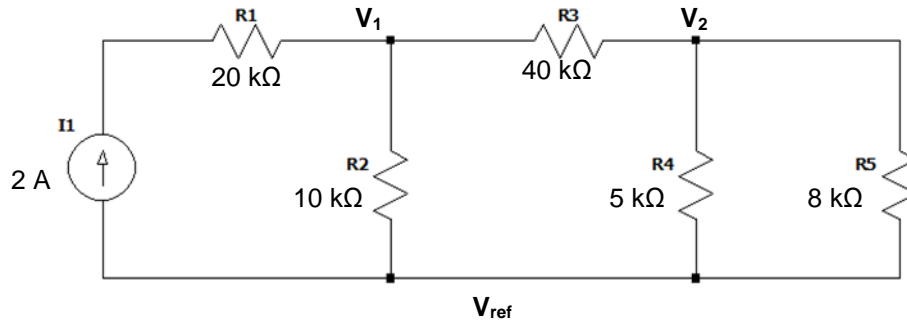
### Example C - Kirchhoff's Current Law (KCL) Application

Find the voltage across  $R_3$  in the following circuit using KCL.

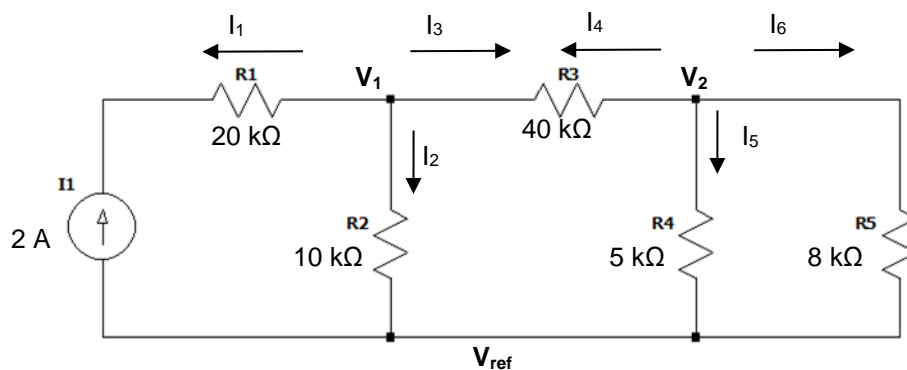


**Solution**

1) Identify the nodes (only essential nodes)



2 & 3) Draw current arrows & Apply KCL (in future, this step may be done mentally and not written down)



3 essential nodes so we need to write (3-1) equations ( $V_1$  and  $V_2$ ):

$$\text{KCL @ } V_1 \rightarrow i_1 + i_2 + i_3 = 0$$

$$\text{KCL @ } V_2 \rightarrow i_4 + i_5 + i_6 = 0$$

4) Apply Ohm's Law and use known values to write KCL equation(s) in term of node voltages & Known currents.

$$\text{KCL @ } V_1 \rightarrow -2 + \frac{V_1}{10,000} + \frac{V_1 - V_2}{40,000} = 0$$

$$\text{KCL @ } V_2 \rightarrow \frac{V_2 - V_1}{40,000} + \frac{V_2}{5,000} + \frac{V_2}{8,000} = 0$$

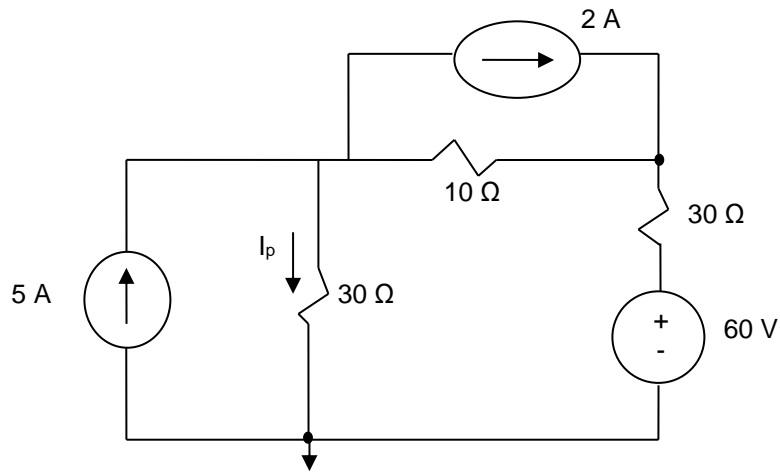
Solving the system of 2 equations  $\rightarrow V_1 = 16.23 \text{ kV}$  and  $V_2 = 1.16 \text{ kV}$

The problem was asking for the voltage across  $R_3$  which can be written as:

$$V_{R_3} = V_1 - V_2 = 16.23 - 1.16 = 15.07 \text{ kV}$$

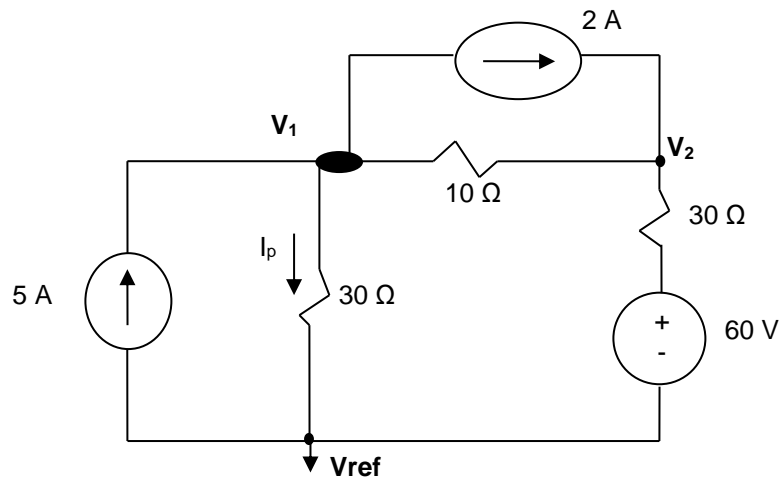
**Example D** -- Kirchhoff's Current Law (KCL) Application

Find  $I_p$  in the following circuit using KCL:

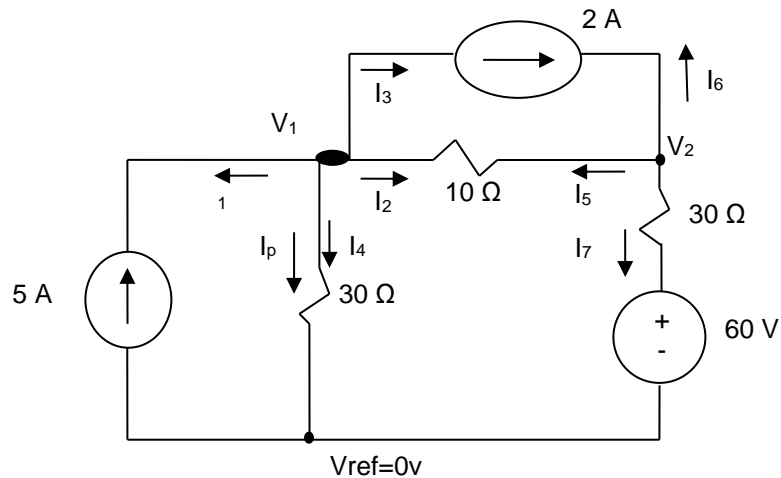


**Solution:**

- 1) Identify the nodes (only essential nodes)



- 2 & 3) Draw current arrows & Apply KCL ((in future, this step may be done mentally and not written down))



$$\begin{aligned} \text{KCL \& } V_1 &\rightarrow I_1 + I_2 + I_3 + I_4 = 0 \\ \text{KCL \& } V_2 &\rightarrow I_5 + I_6 + I_7 = 0 \end{aligned}$$

4) Apply Ohm's Law and use known values to write KCL equation(s) in term of Node voltages

$$\begin{aligned} I_1 &= -5 \text{ A} \\ I_2 &= (V_1 - V_2)/10 \\ I_3 &= 2 \text{ A} \\ I_4 &= V_1/30 \\ I_5 &= (V_2 - V_1)/10 \\ I_6 &= -2 \text{ A} \\ I_7 &= (V_2 - 60)/30 \end{aligned}$$

Apply the known values to earlier KCL equations for Node 1 & 2:

$$\begin{aligned} \text{KCL \& } V_1 &\rightarrow -5 + (V_1 - V_2)/10 + 2 + V_1/30 = 0 \\ \text{KCL \& } V_2 &\rightarrow (V_2 - V_1)/10 - 2 + (V_2 - 60)/30 = 0 \end{aligned}$$

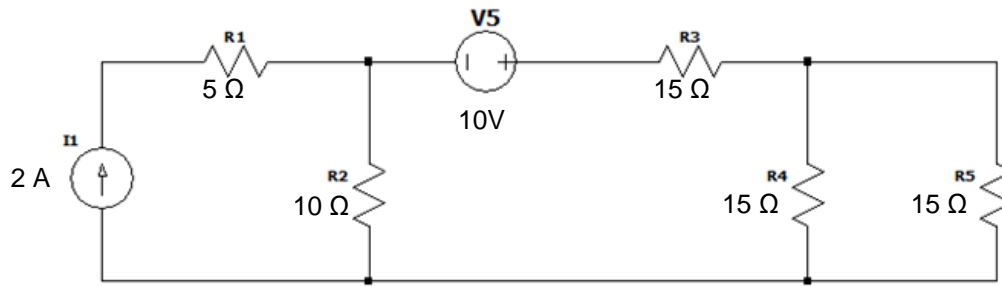
Simplify the above two simultaneous equations:

$$\begin{aligned} \text{KCL \& } V_1 &\rightarrow 4V_1 - 3V_2 = 90 \\ \text{KCL \& } V_2 &\rightarrow -3V_1 + 4V_2 = 120 \end{aligned}$$

Solve the above two simultaneous equations to find  $V_1$ . Note that most scientific calculators have matrix functionality which is able to solve the multiple simultaneous equations.

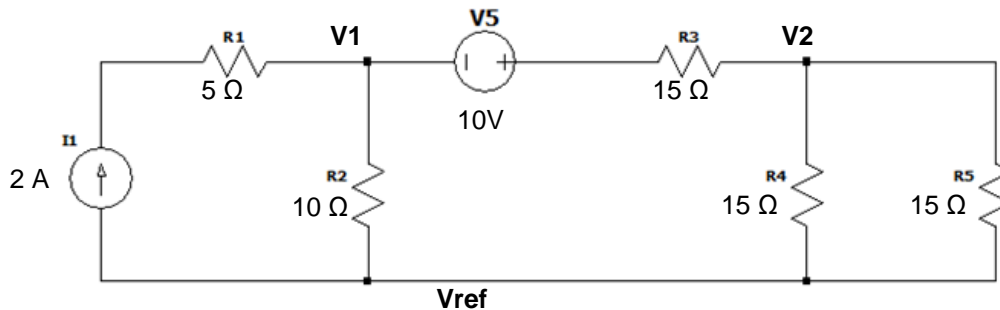
$$V_1 = 102.9 \text{ V} \rightarrow I_p = I_4 = V_1/30 = 3.4 \text{ A}$$

**Example F-** Kirchoff's Current Law (KCL) Application  
Find Voltage across R2 using KCL.

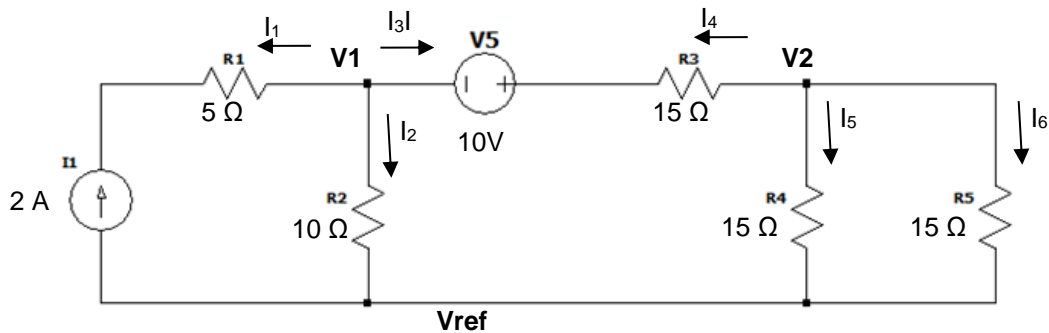


**Solution**

1) Identify the nodes (only essential nodes)



2 & 3) Draw current arrows & Apply KCL (this step may be done mentally and not written down)



$$\begin{aligned} \text{KCL \& } V_1 &\rightarrow I_1 + I_2 + I_3 = 0 \\ \text{KCL \& } V_2 &\rightarrow I_4 + I_5 + I_6 + I_7 = 0 \end{aligned}$$

4) Apply Ohm's Law and use known values to write KCL equation(s) in term of Node voltages and known currents.

$$\text{KCL \& } V_1 \rightarrow -2A + \frac{V_1}{10\Omega} + \frac{V_1 - (-10V) - V_2}{15\Omega} = 0$$

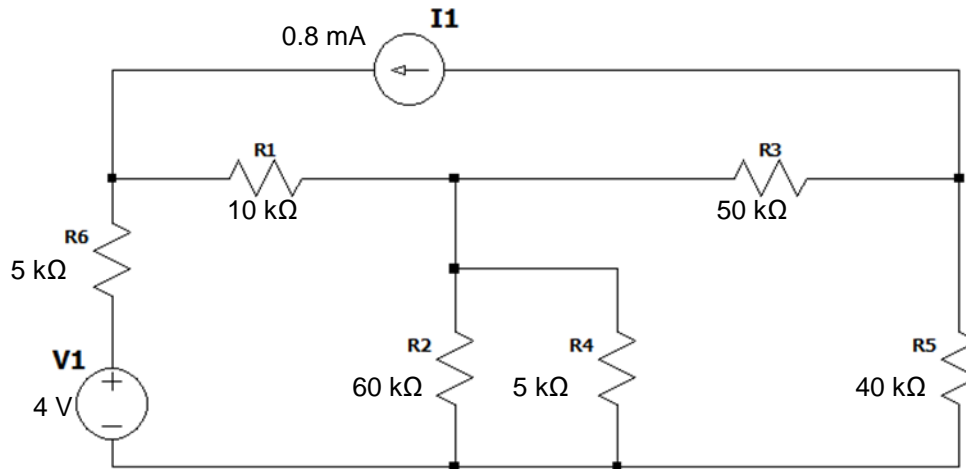
$$\text{KCL \& } V_2 \rightarrow \frac{V_2 - (+10V) - V_1}{15\Omega} + \frac{V_2}{15\Omega} + \frac{V_2}{15\Omega} = 0$$

Solve the system of equations  $\rightarrow V_1 = 10.77 \text{ V}, V_2 = 6.92 \text{ V}$

The problem was asking for voltage across R2 which is V1 therefore  $V_{R2} = V_1 = 10.77 \text{ V}$

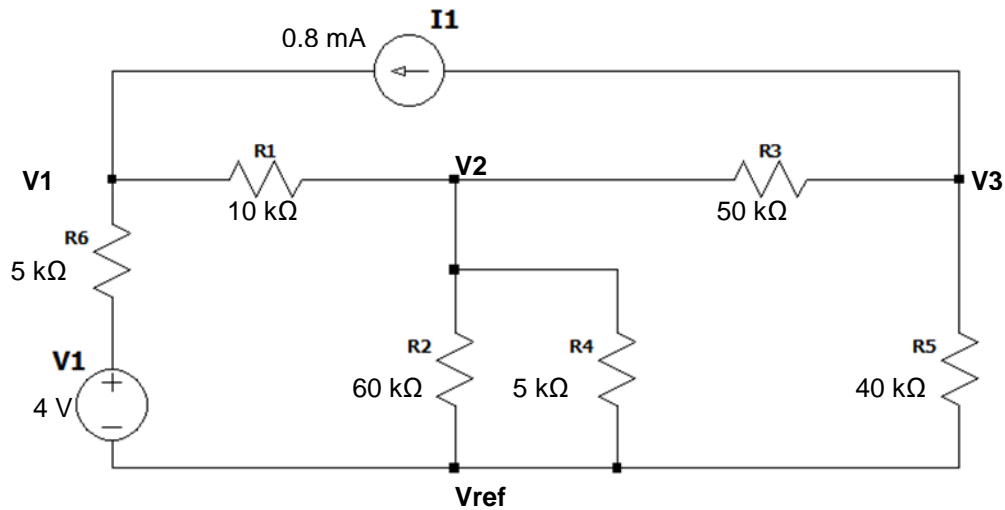
**Example G** - Kirchhoff's Current Law (KCL) Application

Find voltages are all essential nodes of the following circuit using KCL.



**Solution**

1) Identify the nodes (only essential nodes)



Now, that we have completed multiple analysis using KCL, we can combine steps 2, 3 & 4. For each node simply for each branch write the outgoing current in term of node voltages or the value of current if it is known. Here is the equation for the three nodes in this circuit and ignoring Vref:

$$\text{KCL at } V1 \rightarrow -0.0008 + \frac{V1-V2}{10,000} + \frac{V1-4}{5,000} = 0$$

$$\text{KCL at } V2 \rightarrow \frac{V2-V1}{10,000} + \frac{V2}{60,000} + \frac{V2}{5,000} + \frac{V2-V3}{50,000} = 0$$

$$\text{KCL at } V_3 \rightarrow 0.0008 + \frac{V_3 - V_2}{50,000} + \frac{V_3}{40,000} = 0$$

Solve the system of 3 equations to find the voltage at all three essential nodes:

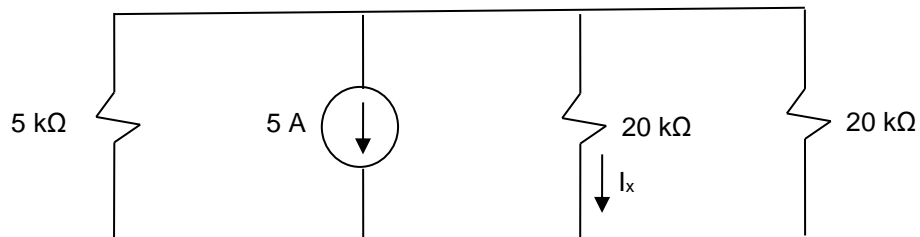
$$V_1 = 5.53 \text{ V}$$

$$V_2 = 0.60 \text{ V}$$

$$V_3 = -17.51$$

**Student Exercise A** - Kirchhoff's Current Law (KCL) Application

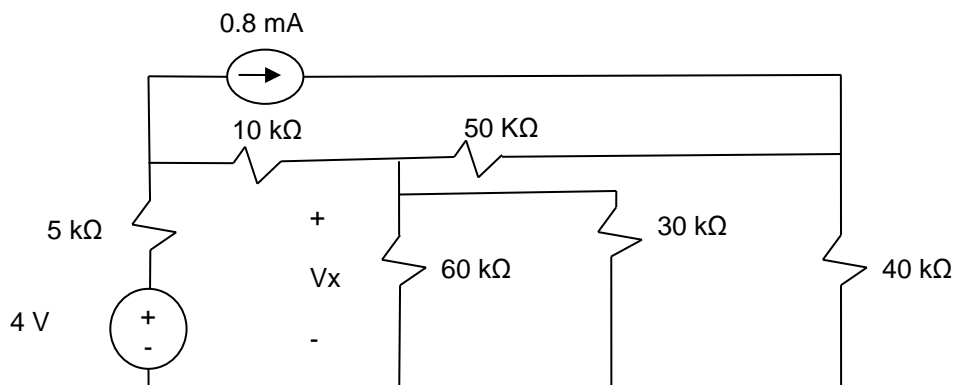
Use KCL to find the value of  $i_x$  in the following circuit:



**Solution**

**Student Exercise B** - Kirchhoff's Current Law (KCL) Application

Use KCL to find the value of  $V_x$  in the following circuit:



Solution



#### 4.8. Additional Resources

Nilsson, J. Electrical Circuits. (2004) Pearson.

#### 4.9. Problems

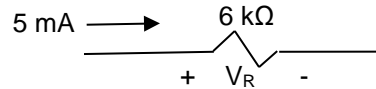
1. Show the relationship between:

- Charge and Current
- Charge and Voltage

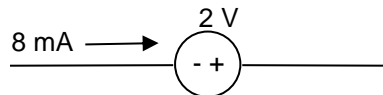
2. What are the three Ohms Law algebraic expressions?

3. Describe the three assumptions of Ideal Circuit Model.

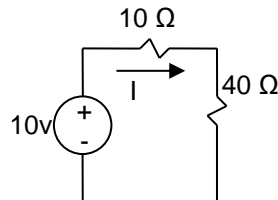
4. Calculate the power in the following resistor:



5. Calculate the power in the following Voltage Source:

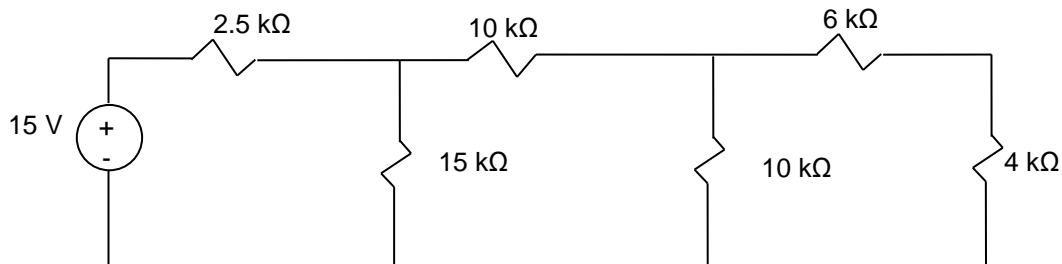


6. For the following circuit:



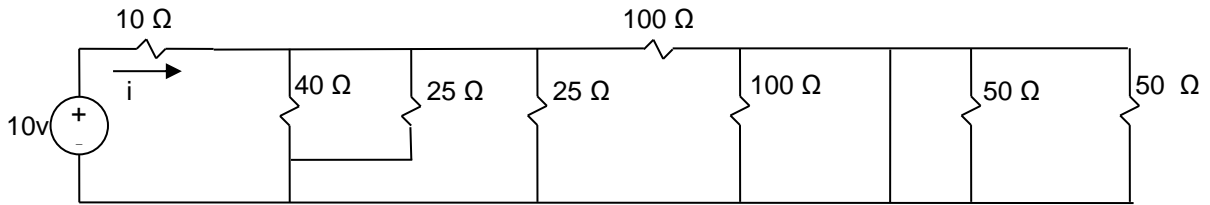
- Calculate the value of  $I$
- What are the quantities of power delivered to the each of the resistors, and the power generated by the voltage source?
- Is this a valid circuit? Show your reasoning.

7. For the following circuit:

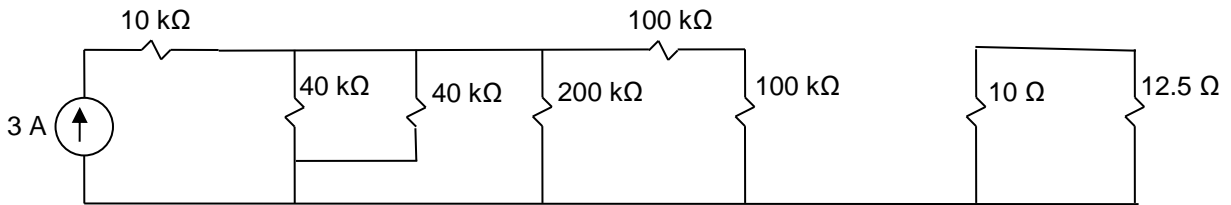


- Simplify the circuit to an equivalent circuit with one resistor and one voltage source.
- Use the resulting circuit from part (a) to find the current through the 15 volt source.
- What is the power delivered by the 15 volt source.

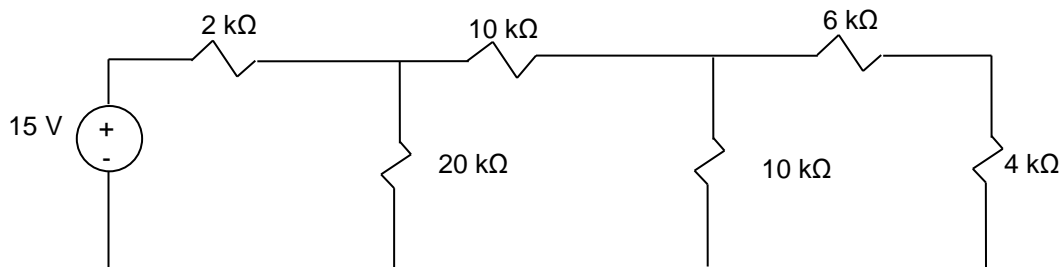
8. Use resistor simplification to find the current through the 10 Ω resistor.



9. Use resistor simplification to find the power through the 3 A current source.

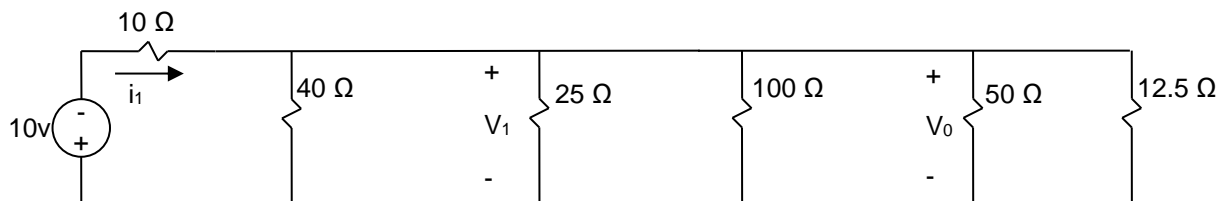


10. For the following circuit:



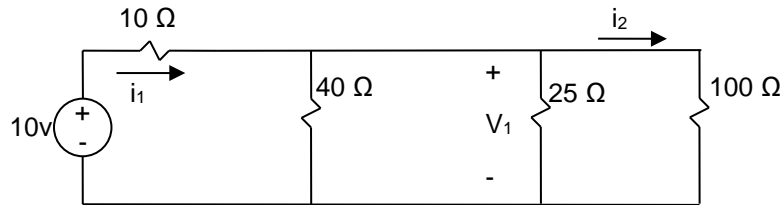
- Identify and count the number of essential nodes.
- How many independent equations can you write using the KCL approach?
- Use KCL to find the current through the 6 kΩ resistor.

11. For the following circuit:

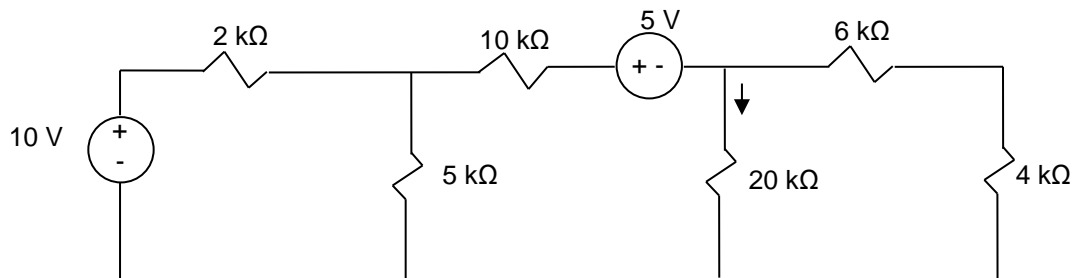


- Identify and count the number of essential nodes.
- How many independent equations can you write using the KCL approach?
- Use KCL to find the values of  $i_1$ ,  $V_1$  and  $V_0$ .

12. Use KCL to calculate the value of  $i_1$ ,  $V_1$  and  $i_2$  in the following circuit:

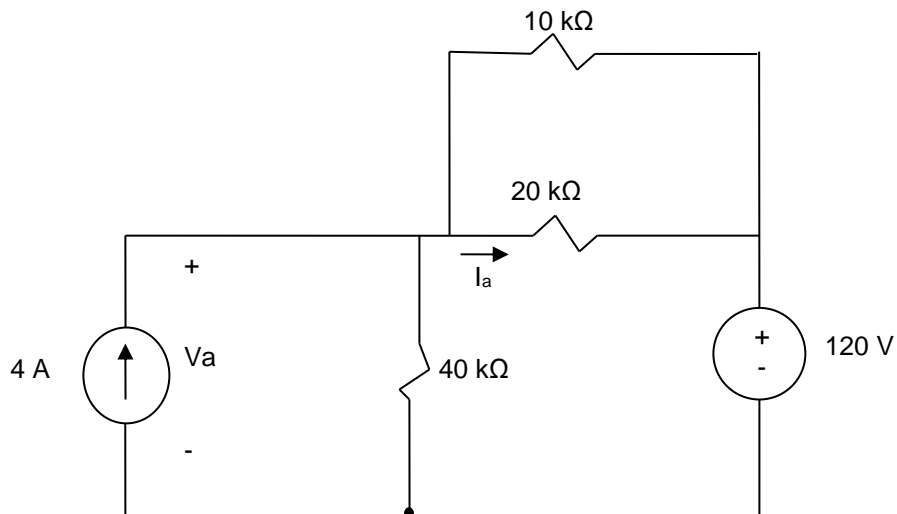


13. For the following circuit:



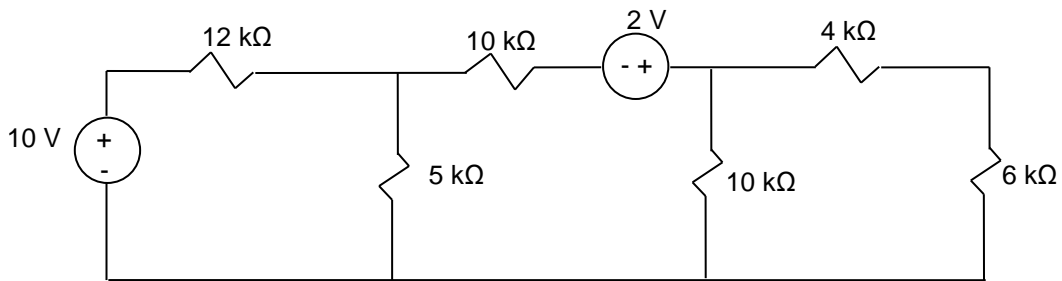
Find the current through the 10 kΩ resistor using KCL.

14. For the following circuit:

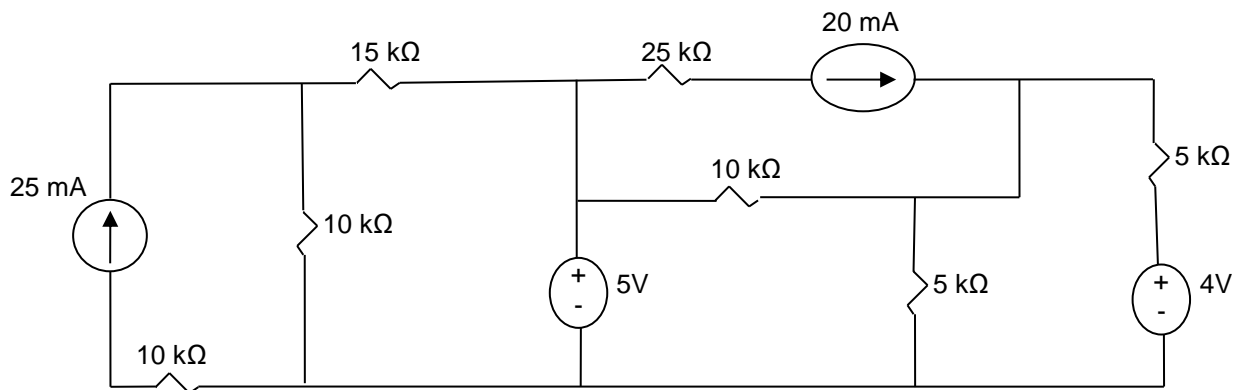


Find  $I_a$  and  $V_a$  using the KCL.

15. Use KCL to calculate the voltage across the 5 kΩ resistor and the power consumed by the 6 kΩ resistor.



16. For the following circuit:



Identify the node with the highest voltage. Show your work.

## Chapter 5. Digital Logic

### 5.1. Key Concepts and Overview

- Digital vs. Analog
- Digital Design Overview
- Binary Number Systems
- Standard Logic Gates & Binary Algebra
- Input and Output Configurations
- Introduction to Logic Design

## 5.2. Digital vs. Analog

Analog or continuous data have continuous values similar to real numbers. On the other hand, digital or discrete data only have distinct value similar to integer numbers. When we want to use a computer to analyze real-world data, the real-world analog data must be converted to computer-legible digital data. A continuous value must be converted to a two-value (binary) system. The following two rules enable the conversion:

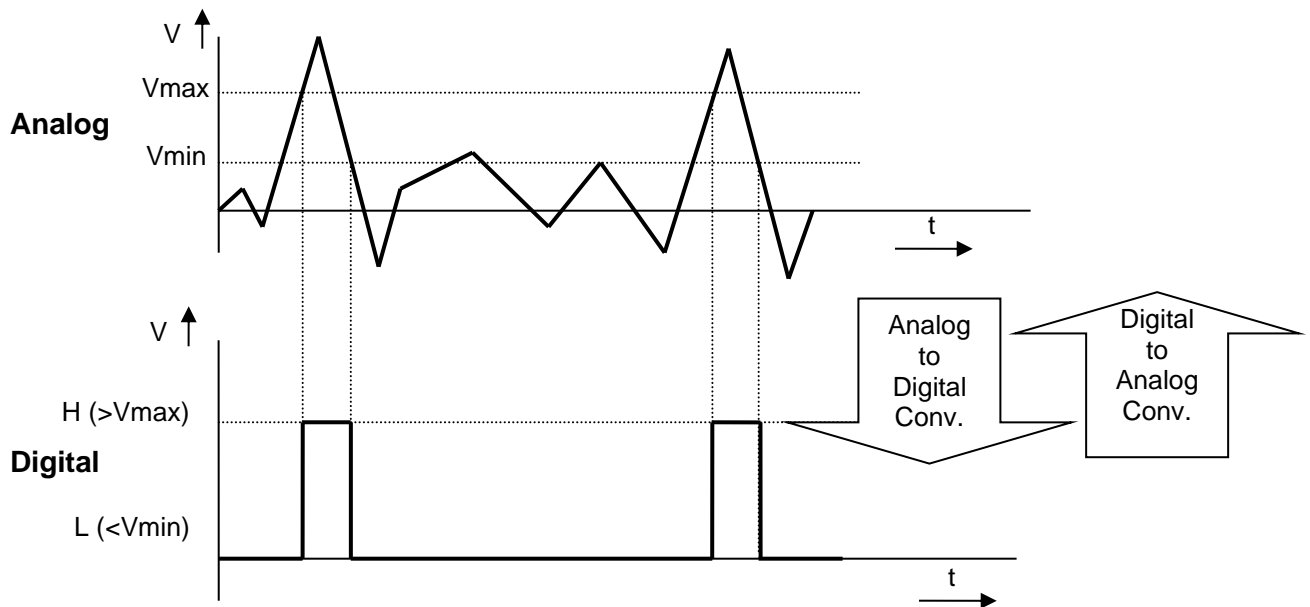
- Above a certain voltage level, the binary value is considered “On”, “High”, “1-state” or “True” (H).

*When  $V > V_{max}$ , it is said to be H*

- Below a certain voltage level, the binary value is considered “Off”, “Low”, “0-state” or “False” (L)

• When  $(V < V_{min})$  it is said to be L.

Here is an example of using the above definition to convert analog signals (heart beat) to digital signal (H & L)



Once the signals are digital, we are able to use logic to build systems that take input and produce output. The first step in the process is being able to represent real world input and output in terms of 0 “L” and 1 “H”.

❖ **Example** - Describe the input and output of a traffic intersection in digital form.

### Solution

Input:

Presence of Car at the intersection:

- \* Car is **present** → digital input value **1**
- \* Car is **not present** → digital input value is **0**

Status of traffic lights

- \* Red light **on** → digital output value is **1**

\* Red light **off** → digital output value is **0**

**Student Exercise – Introduction**

What are some examples of digital systems? Identify input and output for these systems.

**Solution**



### 5.3. Digital Design Overview

This Section introduces the idea of Digital Design from transistors to computer systems. All digital systems from smallest to largest run on a 2-valued system (commonly referred to as binary or digital system), so an electronic solution is needed to represent the two values. This is typically accomplished with a switch that can be on or off. In the early days, mechanical switches were used, followed by vacuum tubes.

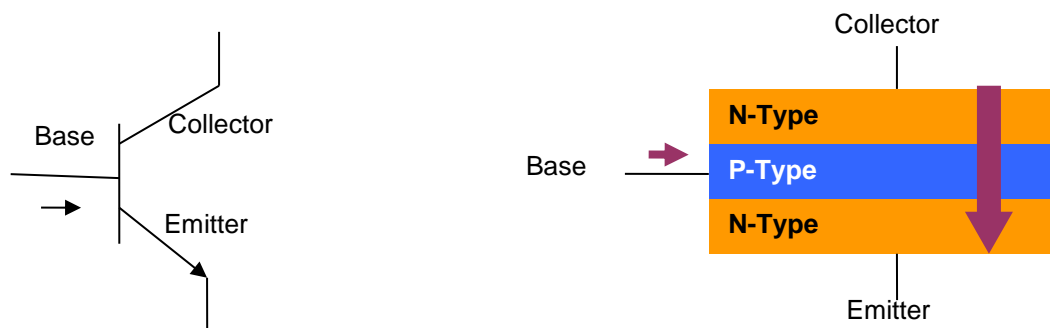
Today, we use transistors that can be configured to approximate the switch on and off modes to build Digital Systems. Transistors are fast (gigahertz –switch billions of times per second), low cost (billions for a few dollars) and small (billions per cm<sup>2</sup>). Transistors are the enablers of electronic and computer revolution.

Here is an overview of transistors and their operation:

- Transistor was invented by three scientists at the Bell Laboratories in 1947 and rapidly replaced vacuum tube as an electronic signal regulator switch.
- Transistor is the basic elements in integrated circuits (ICs), which consist of very large numbers of transistors interconnected with circuitry and packaged into a single silicon microchip or "chip." A typical processor chip has billions of transistors.
- Transistor is most commonly built on semiconductor material and the ones design for digital use are a switch (close or open):

#### Example - NPN Bipolar Transistor

NPN bipolar transistor was one of the earliest transistor designs. Its operation consists of allowing current from collector to emitter or not (switch close and open) as shown below:



The semiconductor material is given special properties by chemical processes. The doping process adds extra electrons to the material (which is then called N-type for the extra negative charge carriers) or creates "holes" in the material's crystal structure (which is then called P-type because it results in more positive charge carriers).

Today's computers use circuitry implemented using Complementary Metal Oxide Semiconductor (CMOS) technology. The advantage of CMOS is that it uses significantly less power than earlier technology when not switching.

## 5.4. Binary Number Systems

This section introduces numbering system with focus on base 2 (binary) numbers. Its conversion to/from base 10 (decimal) and base 16 (Hexadecimal).

### Decimal Numbers (base or radix 10)

Humans use the decimal numbering system as a default, so when you see the number 56, your assumption is that its base or radix is 10 or  $(56)_{10}$  which is "56 base 10". Each digit is weighted by a power of 10, based on its position in the sequence from the least significant digit (LSD, power of 0) to the most significant digit (MSD, highest power). Each digit must be between 0 and 9 (less than 10).

For example,  $(2375.46)_{10}$  is evaluated as:

	MSD			LSD		
Digit notation	$d_3$	$d_2$	$d_1$	$d_0$	$d_{-1}$	$d_{-2}$
Digit	2	3	7	5	4	6
Value	$10^3$	$10^2$	$10^1$	$10^0$	$10^{-1}$	$10^{-2}$
Results=Value*Digit	2000	300	70	5	0.4	0.06

$$(2375.46)_{10} = 2000 + 300 + 70 + 5 + 0.4 + 0.06$$

Note: The general term for a decimal point is *radix point*

### Binary Number (Base or radix 2)

Computer technology is based on the binary number system, since transistors only have two positions: on or off (1, 0). Each digit of a binary number is called a bit, which is shorthand for binary digits. Also, a group of 8 bits is called a byte and a group of 4 bits is referred to as a nibble. Each bit is weighted by a power of 2, based on its position in the sequence from the least significant bit (LSB) to the Most significant bit (MSB). Each bit must be either 0 or 1 (less than 2).

For example,  $(1010.11)_2$  is evaluated as:

	MSB			LSB		
Digit notation	$b_3$	$b_2$	$b_1$	$b_0$	$b_{-1}$	$b_{-2}$
Digit	1	0	1	0	1	1
Value	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$
Results=Value*Digit	8	0	2	0	0.5	0.25

$$(1010.11)_2 = 8 + 0 + 2 + 0 + 0.5 + 0.25 = (10.75)_{10}$$

Note: The general term for a decimal point is radix point

In Binary, the count starts at 0 (called 0-referencing) where as in Decimal, the count typically starts with 1 (called 1-referencing)

Now that we have two numbering system, it is time to learn to convert from to another.

### Decimal to Binary Conversion – Subtract the weight method

Here are the steps to convert a decimal number to a Binary number:

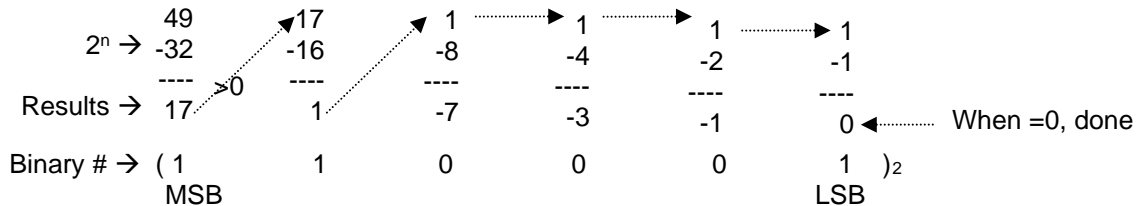
- 1) Find the largest power of 2 ( $2^n$ ) that can be subtracted out of the decimal number. Subtract the power of 2 from the number and write a "1".
- 2) Take the result and subtract ( $2^{n-1}$ )
  - a. If the result is positive, then that bit is one

- b. If the result is negative, then that bit is zero and the result equals the result from step 1.  
 3) Repeat step 2 until the result is exactly 0.

**Example – Decimal to Binary Conversion**

Convert  $(49)_{10}$  to a binary number

**Solution**



**Binary to Decimal Conversion – “Add the weight method”**

Converting from Binary to Decimal is easier than Decimal to Binary conversion in previous section. To Convert a binary number to its equivalent number in decimals, simply multiply each bit with its weight and add to get the decimal number.

**Example – Decimal to Binary Conversion**

Convert  $(110001)_2$  to a decimal number.

**Solution**

$$(110001)_2 = (1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0)_{10} = (49)_{10}$$

**Hexadecimal Number (base 16 or Hex)**

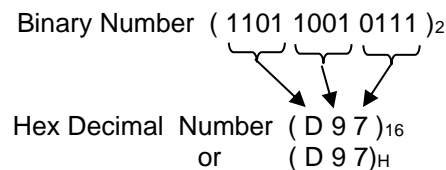
Hexadecimal numbers are used by humans to condense binary numbers to smaller number of digit for ease of reading and communicating digital data. Hex digits can be from 0 to 9 and A-F (representing 10-15).

The reason Hexadecimal system is popular is due to its ease of conversion from and to binary numbers. To convert a binary number to hex, from the right hand side, group bits into sets of 4 and convert each set to hex digit.

**Example – Binary to Hex Conversion**

Convert  $( 1101 1001 0111 )_2$  to its hexadecimal equivalent.

**Solution**



As you can see D97 is easier to communicate than 110110010111 and less error prone.

**Binary Codes**

Information has to be translated into binary in order to be processed by digital systems. Some of the most commonly used binary codes are the ones that translate language characters to binary code.

American Standard Code for Information Interchange Code (ASCII) was used represent alphanumeric and control characters in 8-bit (256 possibilities). Unicode is a 16-bit derivative of ASCII code that is implemented in most of today's computers. With 16 bits, Unicode allows for more than 65,536 characters which are enough to represent all of the world's language characters.

Here is ASCII table show each character and its equivalent code:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: [www.asciitable.com](http://www.asciitable.com)

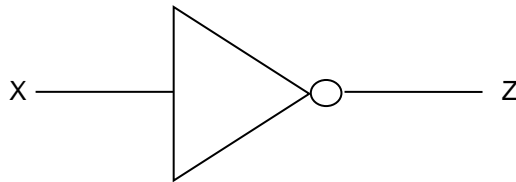
## 5.5. Standard Logic Gates & Binary Algebra

This section introduces logic gates which perform basic logic operations such as “AND”, “OR” and “NOT” operations on one bit binary numbers (0 and 1). Later in the section, we will use these logic gates to develop expressions and logic systems.

### “NOT” or “Inverter” Logic gate

“NOT” gate output is inversion (not) of input. So if input is “1” then output is “0” and vice versa.

- “NOT” Binary Algebraic Expression  $\rightarrow Z = \overline{X} = \sim X = X'$
- “NOT” Schematic Symbol



- “NOT” Truth Table

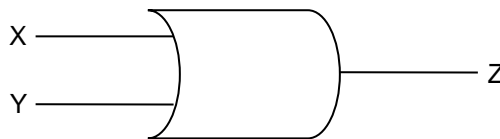
X	Z=X'
0	1
1	0

- “NOT” Chip  
74LS04 chip includes 6 “NOT” gates. Refer to Data Sheet at [EngrCS.com](http://EngrCS.com) for detailed description and usage. *LS in the part number refers to “Low-Power Schottkey” technology; 74HC04 is another version that uses “High Speed CMOS” technology.*

### “OR” Logic gate

If either input to OR gate is “1” then output is “1”, otherwise output is “0”.

- “OR” Binary Algebraic Expression  $\rightarrow Z = X + Y$
- “OR” Schematic Symbol



- “OR” Truth Table

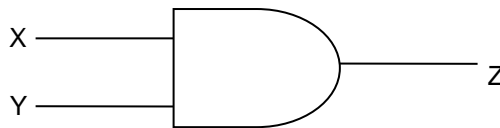
X	Y	Z=X+Y
0	0	0
0	1	1
1	0	1
1	1	1

- “OR” Chip  
74LS32 chip includes 4 “OR” gates. Refer to Data Sheet at EngrCS.com for detailed description and usage. *LS in the part number refers to “Low-Power Schottkey” technology; 74HC32 is another version that uses “High Speed CMOS” technology.*

### “AND” Logic gate

If both input to “AND” gate are “1” then output is “1” otherwise out is “0”.

- “AND” Binary Algebraic Expression  $\rightarrow Z = X \cdot Y$
- “AND” Schematic Symbol



- “AND” Truth Table

X	Y	Z=X+Y
0	0	0
0	1	0
1	0	0
1	1	1

- “AND” Chip  
74LS08 chip includes 4 “OR” gates. Refer to Data Sheet at EngrCS.com for detailed description and usage. *LS in the part number refers to “Low-Power Schottkey” technology; 74HC08 is another version that uses “High Speed CMOS” technology.*

### “XOR” Exclusive OR Logic Gate

If the two XOR gate input are different then output is “1”, otherwise output is “0”.

- “XOR” Binary Algebraic Expression  $\rightarrow Z = X \text{ (Xor) } Y$
- “OR” Schematic Symbol



- “XOR” Truth Table

X	Y	Z=X+Y
0	0	0
0	1	1

1	0	1
1	1	0

- “OR” Chip  
74LS86 chip includes 4 “XOR” gates. Refer to Data Sheet at EngrCS.com for detailed description and usage. *LS in the part number refers to “Low-Power Schottkey” technology; 74HC86 is another version that uses “High Speed CMOS” technology.*

The 4 logic gates introduced so far are the basic gates that provide a complete set of logic operators to design all other logical functions. There are a large number of logic chips with more complex operation. Date Sheets with detailed information for some are available at engrcs.com.

### Binary Algebraic Expressions

Binary algebraic expressions use basic logic operators discussed earlier (“AND”, “OR” and “NOT”) to describe more complex systems. Below is an expression describing a system where X, Y and Z represent the input and F is the output.

$$F(X,Y,Z) = Z + X \cdot Y + X$$

Just like algebra, we can find values of F for every possible input by plugging every possible value of input in the expression and evaluate them to find the out. The good news is that binary input can only be “0” or “1” so the possibilities are much more limited compared to Decimal algebra. Truth Table is a tool to organize all the input and corresponding output as shown below:

#### Input Variables

All possible values of 3 bit binary input (X,Y,Z). It is important to list the input in order from smallest “0” to largest “7” to make sure all possibilities are listed.

#### Output Functions

For each possible input, evaluate the output by plugging input values in expression and evaluating the output function.

X	Y	Z	A = Z + X·Y + X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Truth Table

When using expressions, it is important to pay attention to the order of operations for logic operators. Here are the operators listed in their order of operation from the Highest to lowest order of operations:

- 1) = Equals
- 2) () Parentheses  
Parentheses are used to force the operation order sequence, much like in decimal algebra.
- 3) NOT
- 4) AND
- 5) OR

**Example – Order of Operation**

Are the following two operations equal?

$$A = Z + X \cdot Y + X$$

$$B = (Z + X) \cdot (Y + X)$$

**Solution**

The simplest way to answer the question is to draw a Truth Table and compare output A and B for every possible input.

Input Variables			Output Functions	
X	Y	Z	A	B
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

**Truth Table**

Second row of table where input is “001”, A evaluate to a “1” which “B” evaluates to a “0” therefore the two expressions are not equal.

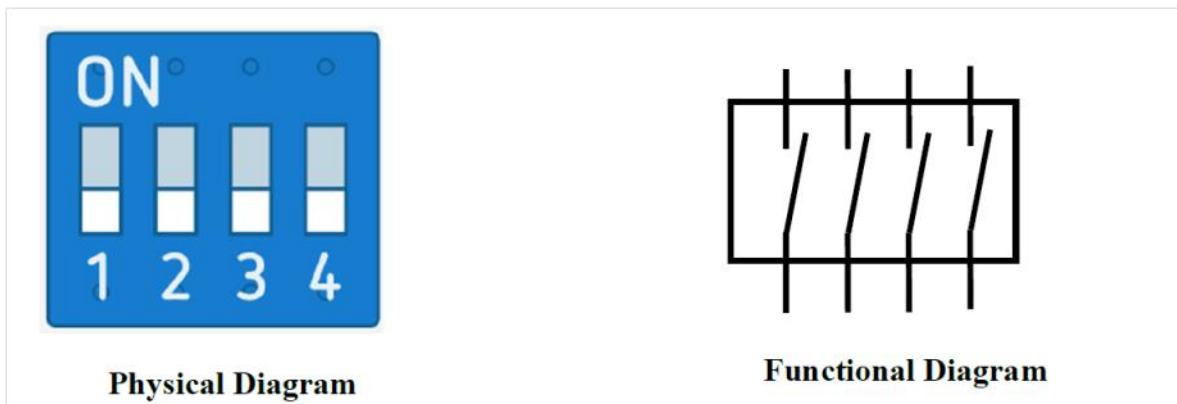


## 5.6. Input and Output Configurations

In order to test any logic circuit, input and output are required. During the development process, switches, and Light Emitting Diodes (LEDs) are used in lab to simulate input and output. This section covers the use of switches to produce “0” or “1” as digital input and the use of LEDs to display “0” and “1” output.

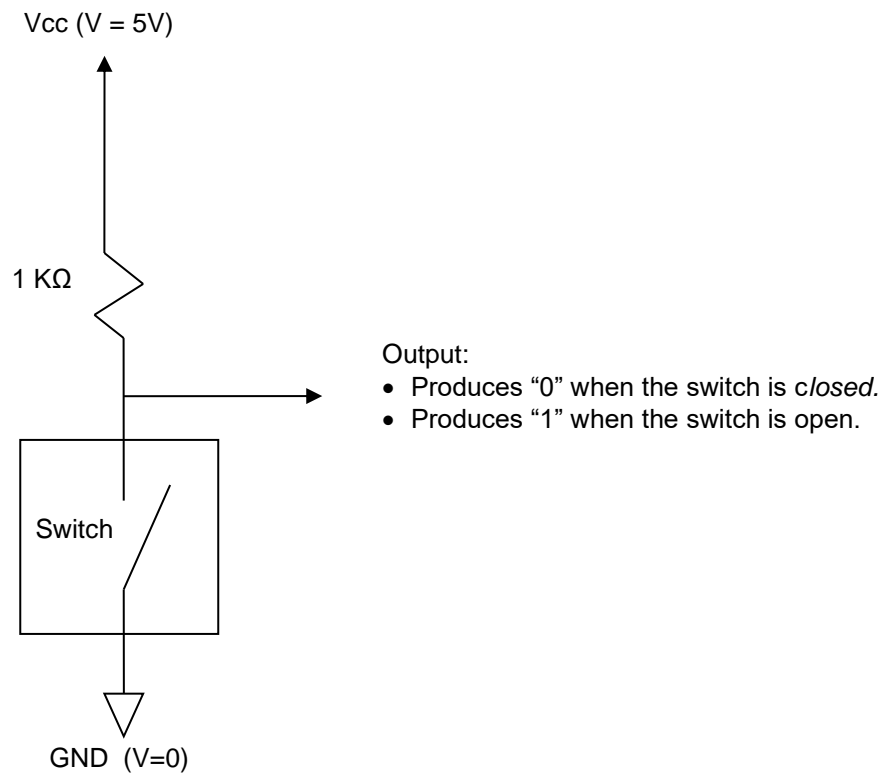
### Simulating Input Using Switches

Switch is either closed (conducting electricity,  $R=0$ ) or open (not conducting electricity,  $R=\infty$ ). Since it is common to have multiple inputs, we typically use switch packs with multiple switches. Below are physical and functional diagram and a 4-pack switch:



*Note: When switch is in “ON” position, the switch is closed ( $R=0$ ).*

The follow switch configuration produces a digital “1” (high Voltage) when the switch is open, and a digital “0” (low Voltage) when the switch is closed.

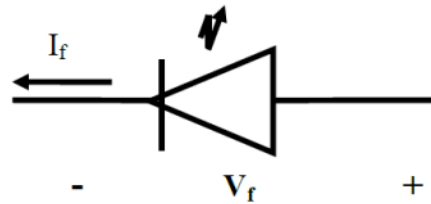


### Light Emitting Diode (LED) as an Output

Light-emitting diodes (LEDs) are used as indicators in many applications, from power on/off lights to traffic signal lights. LED lamination, current and power specifications vary, depending on design and application. One common LED is the SSL-LX5093LXX, manufactured by Lumex (refer to [engracs.com](http://engracs.com) for data sheet):



**Physical Diagram**



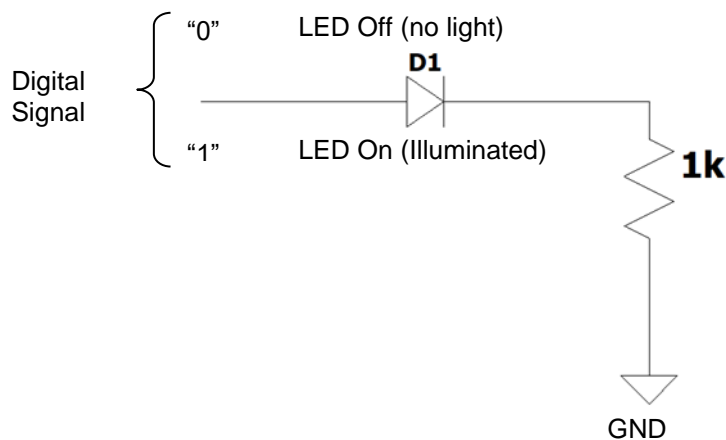
**Functional Diagram**

**Rating:**  $I_f < 30 \text{ mA}$  at 2.5 Volts

**Typical:** +5 V at  $I = 5 \text{ ma}$

*Note: On the Physical Package, negative side is flat (not rounded) and the lead is shorter than the positive side.*

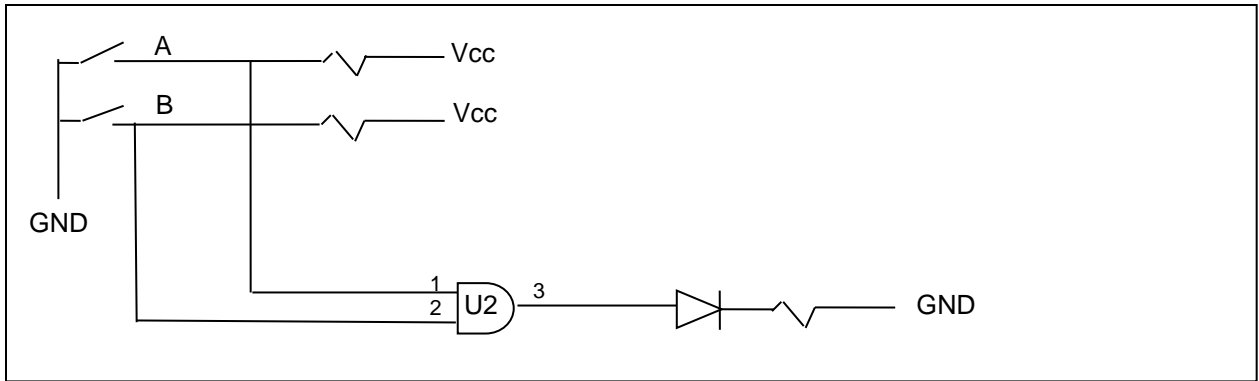
Below is an example of using an LED to display a digital signal value:



Digital Signal may be from your switching show earlier or from the output of one of the logic gates. It is important to add the 1k resistor between LED and ground to limit the current through LED.

**Example** – Simulating Digital Input and Output  
Design a logic circuit to test the functionality of “AND” gate in 74LS08.

**Solution**  
The following schematics test  $S = A \cdot B$ .



**Student Exercise** – Simulating Digital Input and Output  
Design a logic circuit to test the functionality of “OR” gate in 74LS32.

**Solution**

## 5.7. Introduction to Logic Circuit Design

In previous sections binary system, logic gates/operators and truth table has been discussed and used in analyzing the function of logic circuits. This section extends the coverage to designing logic circuits from requirements to schematics through examples.

The logic circuit design process from requirements to schematics consists of 4 steps:

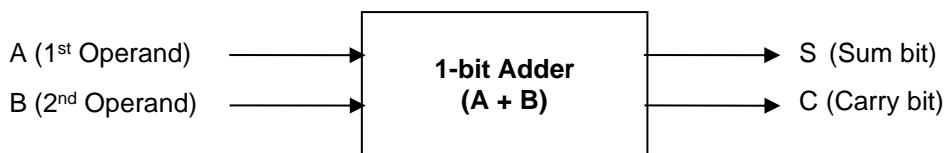
- Step 1.** Understand the Requirement and define input and output along with meaning of “1” and “0” for each input and out. Draw System Diagram (A box with input arrow coming in and output arrows going out).
- Step 2.** Truth Table for the system showing the relationship between input and output.
- Step 3.** Write expression for each of the Output Functions
- Step 4.** Draw a circuit schematic including:
  - Components with pin numbers and part identification
  - Connections between the components using either horizontal or vertical lines
  - Component Block with description of each component
  - Author Block with the name of designer and design date and any revision dates.

### Example – Logic Circuit Design

Design a 1-bit adder circuit that accepts two 1-bit input (A and B) and display the output sum (S) and carry out (C):

### Solution

Step 1. System Diagram (Input & Output Definition)



Step 2. Truth Table (Input and Output Relationships)

Input		Output	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Step 3. Output Function

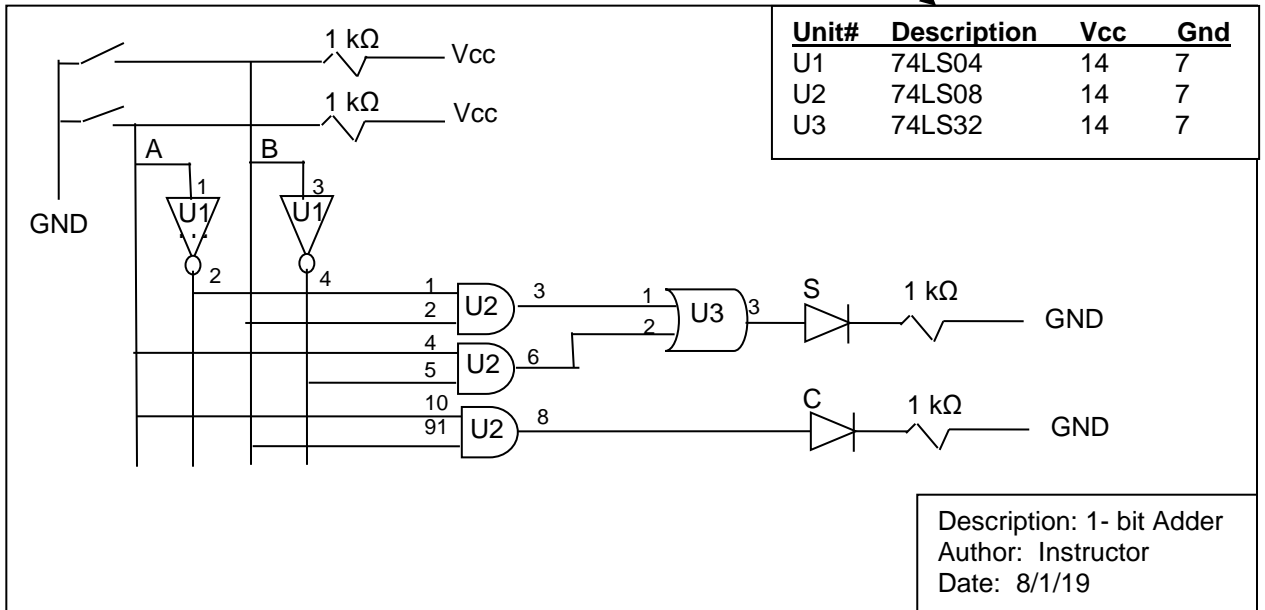
$$S = A'.B + A.B'$$

$$C = A.B$$

Step 4. Schematics (Component Level Diagram)

Note: schematics must be on separate sheet of paper and include project and component identification blocks. Typically, landscape is preferred layout for schematics.

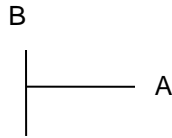
"Component ID Block"



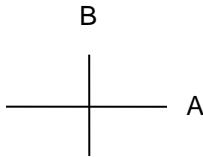
"Project ID Block"

### Wire Connection Convention

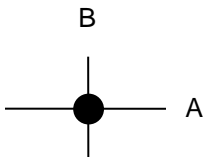
In digital circuit schematics, the following connection convention is used to show if there are connection between wires that intersect on the schematic drawing:



**Tee** – when a line tees into another line, it is to indicate electrical connection between the lines (lines A and B are connected)



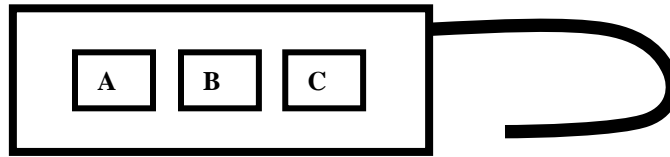
**Crossing** – when a line crosses another line, no electrical connection exist (line A and B are NOT connected)



**Connecting** – when a line crosses another line and has a dot on the crossing point, it is to indicate electrical connection between the lines (lines A and B are connected)

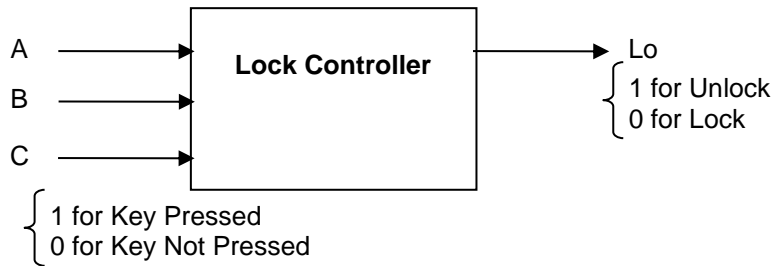
**Example**– Logic Circuit Design

Design a 3-button lock that opens when at least two adjacent buttons are pressed.



**Solution**

Step 1. System Diagram (Input & Output Definition)



Step 2. Truth Table (Input and Output Relationships)

Input			Output
A	B	C	Lo
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Step 3. Output Function

$$Lo = A'BC + ABC' + ABC$$

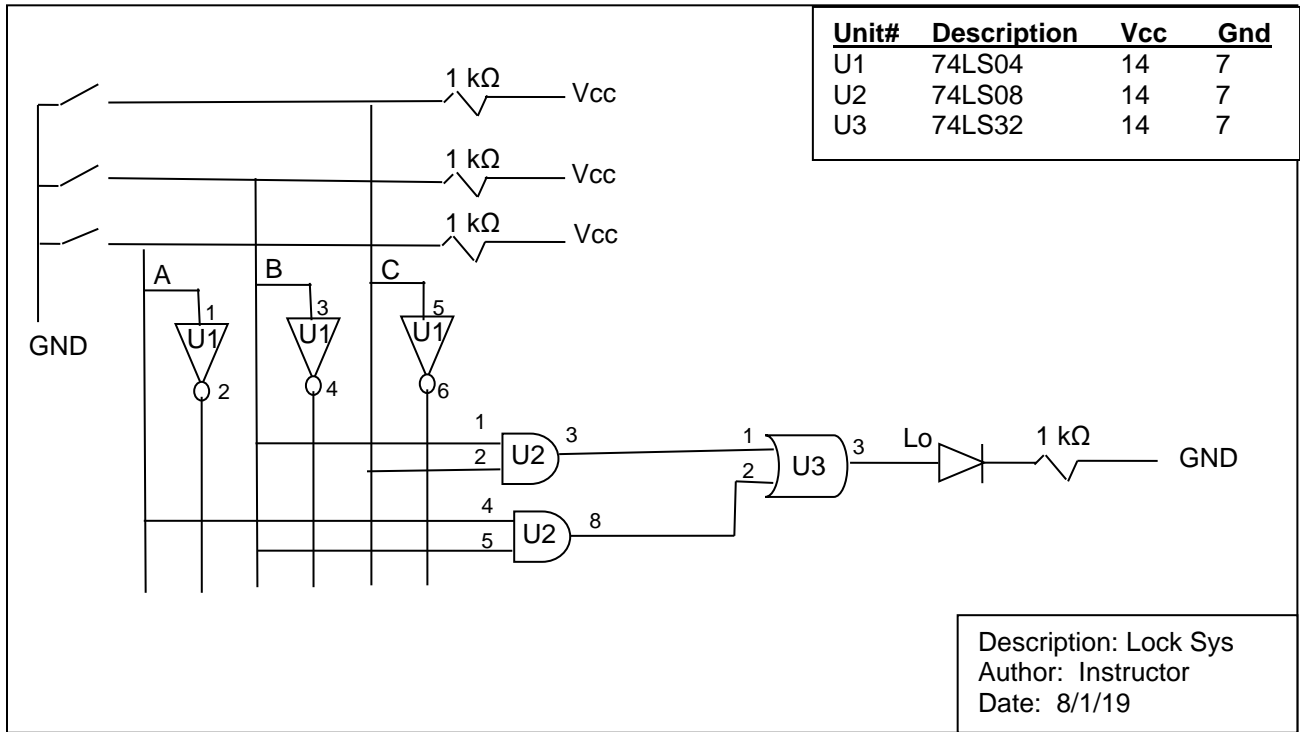
The Adjacency theorem allows for elimination of a variable if the variable and its inversion are present in two otherwise same terms. For example the expression  $A.B + A.B'$  can be simplified to  $A$ .

Applying Adjacency theorem to  $Lo$ , we can write the simplified expression as:

$$Lo = BC + AB$$



**Step 4. Schematics (Component Level Diagram)**



**Student Exercise – Logic Circuit Design**

Design a circuit that accepts two digital inputs ( $A_1$  and  $A_0$ ) and lights up two LEDs ( $L_1$  and  $L_0$ ) as described by the following table:

$A_1$	$A_0$	$L_1$	$L_0$
0	0	On	Off
0	1	On	Off
1	0	Off	Off
1	1	Off	On

**Solution**

Step 1. System Diagram (Input & Output Definition)

Step 2. Truth Table (Input and Output Relationships)

Step 3. Write Output Functions

Step 4. Schematics (Component Level Diagram)

- ❖ **Student Exercise** – Design a circuit that accepts three inputs A, B and C. If  $A = 1$ , turn on LED  $L_a$  regardless of other inputs. LED  $L_b$  will turn on only when  $A=0$  and  $B=1$  regardless of value of C. Finally LED  $L_c$  turns on only if  $A=0$ ,  $B=0$ , and  $C=1$ .

**Solution:** In-Class Exercise

**Step 1. System Diagram (Input & Output Definition)**

**Step 2. Truth Table (Input and Output Relationships)**

**Step 3. Output Function (Apply Adjacency Theorem to Minimize)**

**Step 4. Schematics (Component Level Diagram)**

## 5.8. Additional Resources

- Wakerley, I. Digital Design. (2006) Prentice Hall
- Katz, R. Contemporary Logic Design. (2005) Pearson.
- Sandige, R. Digital Design Essentials. (2002) Prentice Hall.

## 5.9. Problems

---

1. In Digital Systems, what are three different name used to refer to high voltage ( $V > V_{max}$ ) and low voltage ( $V < V_{min}$ )?

---

2. In today's computer processors, which of the following statements are correct:

- a) Circuitry is implement with Metal Oxide Semiconductor (CMOS) Technology
- b) CMOS uses significantly less power than earlier technologies such as Bipolar when not switching
- c) Number of transistors is in millions
- d) Number of transistors is in billions

*Hint: There may more than one correct statement.*

---

3. Convert the following binary numbers to decimal equivalent value. Check your work by converting the number back to its original base after every conversion:

- a)  $(1101)_2$
  - b)  $(101001)_2$
  - c)  $(11001010)_2$
  - d)  $(10001100)_2$
- 

4. Convert the following decimal numbers to binary equivalent value. Check your work by converting the number back to its original base after every conversion:

- a)  $(35)_{10}$
  - b)  $(139)_{10}$
  - c)  $(425)_{10}$
- 

5. Convert the following numbers to equivalent binary and decimal values:

- a)  $(291)_H$
  - b)  $(CAB)_H$
- 

6. Write the Hexadecimal equivalent of the text string "Hello World" in ASCII code.

---

7. Create a truth table for the following binary expression:

$$F(X, Y, Z) = (X + Y + Z).(X.Y' + X'.Y)$$

---

8. Create a truth table for the following two expressions. Use the truth table to determine if they are equal.

$$F_1(A, B, C) = (A.B + A'.C)'$$
$$F_2(A, B, C) = (A' + B').(A + C')$$

---

9. Design a circuit with two switches and two LEDs plus resistors and wires as needed such that changing the switch position turns on and off the corresponding LED.

---

10. Design a 1-bit multiplier that accepts two 1-bit operands from switches and displays the multiplication results in binary using LED (on means 1 and off means 0). For your design include system diagram, truth table, output function expressions and schematics.

---

---

11. Design a lock with 4 keys that opens when exactly two of the four keys are pressed. For your design include system diagram, truth table, output function expressions and schematics.

## **Chapter 6. Computer Architecture and Programming Fundamentals**

### **6.1. Key Concepts and Overview**

- Computer Architecture
- Programming Levels
- Software Development Steps
- Common Programming Languages
- Software Development Steps

## 6.2. Computer Architecture

In computer science, the word **architecture** describes the relationships of major components of a computer much like a home plan describes the architecture of a home. In this section, architecture will be discussed in terms of design considerations, software layers and hardware.

### Computer Design Considerations

Over time, the importance of various design considerations have changed. But three main areas have remained important: speed/performance, maintainability/reliability, and hardware cost/memory requirements.

#### Speed/Performance

Today, performance continues to be important issues as applications have increased in complexity due to increased demands for graphics, database capabilities, and operating systems. Designers have attempted to answer the needs by:

- (1) Increasing processor performance and memory
- (2) Increasing size and types of memory (cache, RAM, secondary storage)
- (3) Using parallel computing and processing
- (4) Improving software and hardware design tools and processes

#### Maintainability/Reliability

As systems become more complicated, the need for maintainability and expandability becomes increasingly more important.

#### Hardware Cost/Memory Requirement

As technology advances, the issue of hardware cost and memory use minimization is becoming less of an issue in designing computers.

A typical desktop computer in 1986 had 512 KB of RAM, where a typical desktop computer in 2020 has 16 GB ( $16 \times 10^9$  bytes) of RAM. This represents a 32,000 fold increase in 34 years.

### Student Exercise – Computer Design

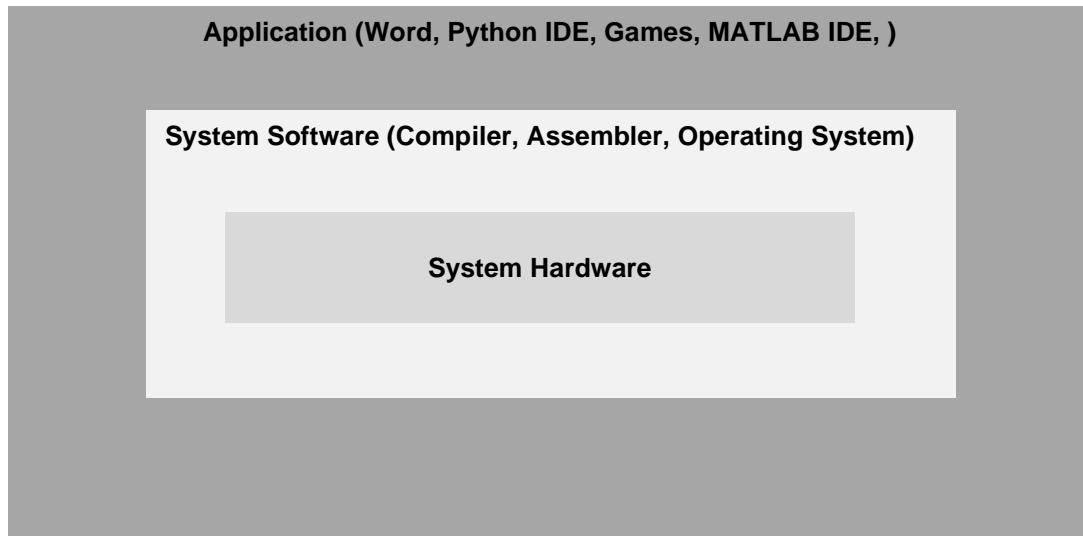
Can you think of another industry or technology other than semiconductors that has shown similar efficiency improvement?

### Solution



### **Computer Software Layers**

Although there are a variety of computer designs and software, we generalize and say that computer software consists of operating system and application software. Below is the visualization of the relationships between hardware and these two layers of software.



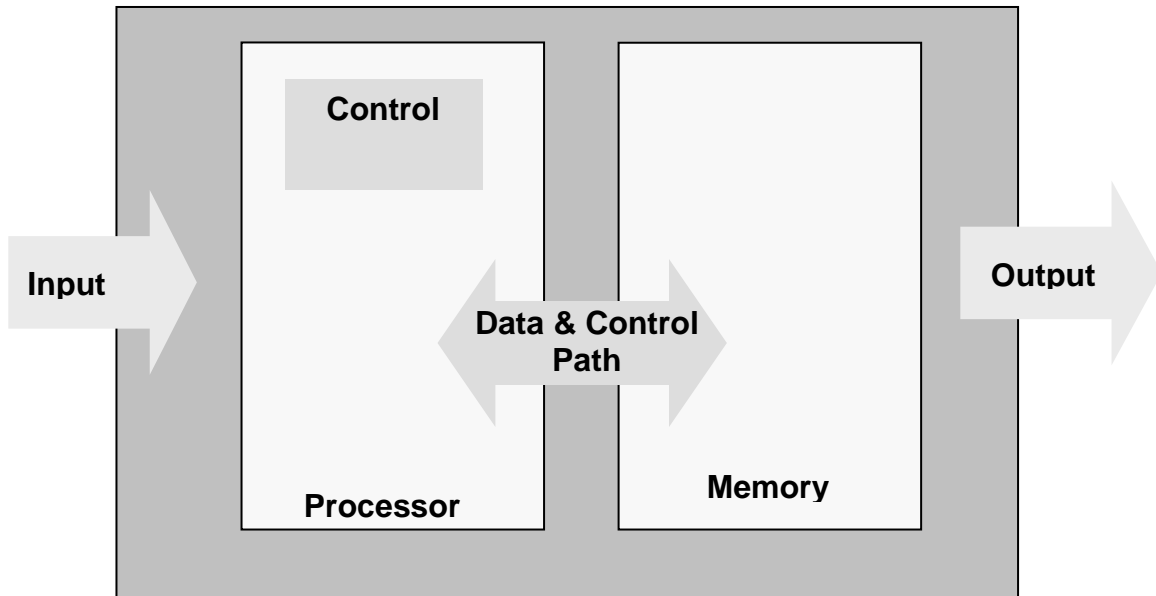
### **Operating System (OS)**

Operating system software and its underlying support are responsible for controlling the functionality and resources of a computer on behalf of users and applications. Microsoft Windows, Linux, and IOS are three of the most popular operating systems in the computer industry. A typical OS delivers the following services:

- input/output management
- Schedule application execution, manage interaction between applications and protect them from each other
- Allocates storage, memory, and processor, and, in general, manages resources among the applications (active processes)

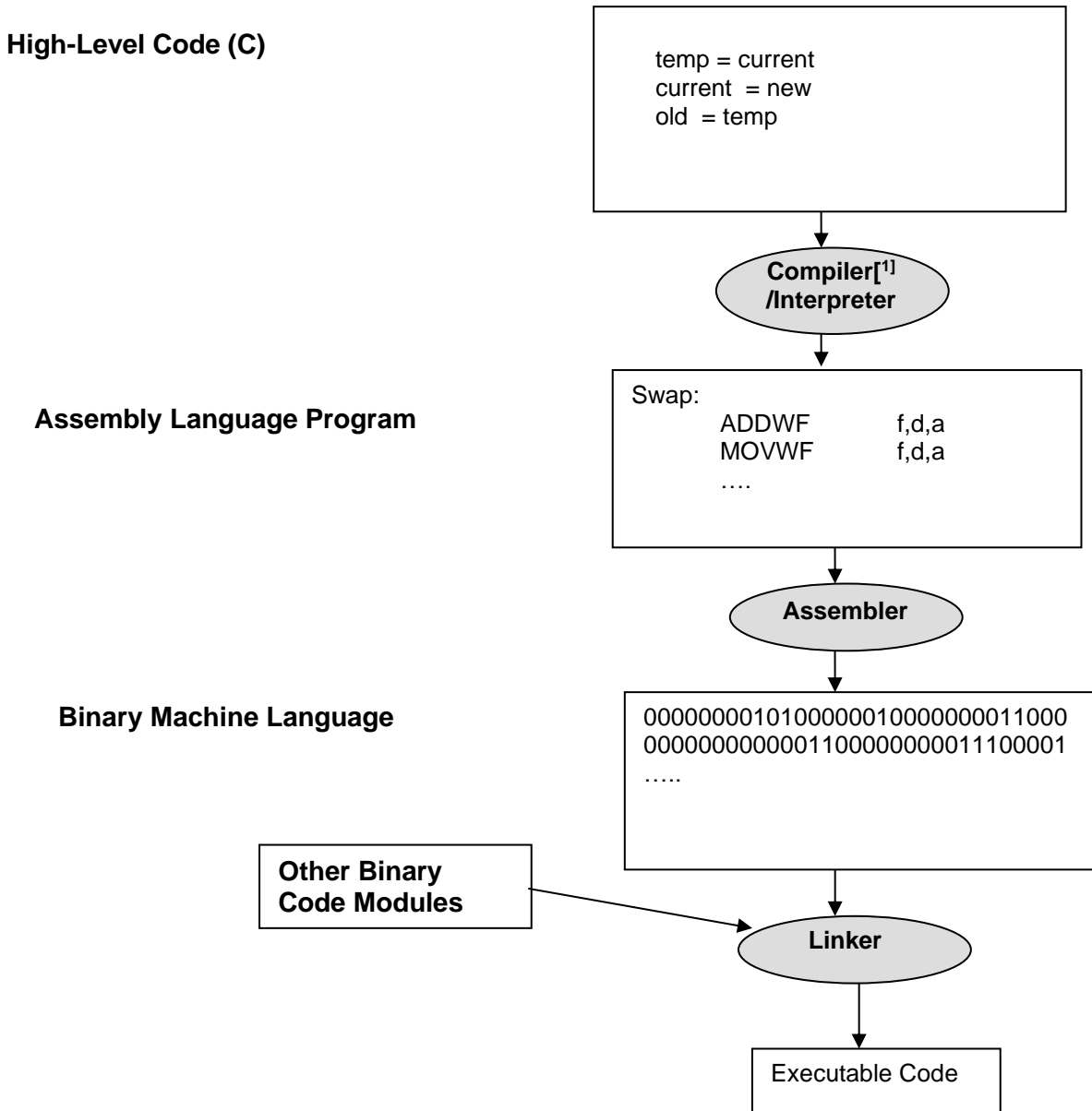
### Computer Hardware Overview

Computer hardware has a Central Processing Unit (CPU) at its core with additional hardware for input, control, memory, execution, and output. The following diagram shows common high level computer hardware architecture:



### 6.3. Programming Levels

Steps from High Level Language (C, C++, Java, Python, MATLAB ...) to executable code include compiling/Interpreter , Assembling and Linking as show below:



**Note:**

[1] Compiler converts high level language into Assembly before execution while interpreter converts high level language into assembly or equivalent during executing. C and C++ are compiled languages while MATLAB and Python are Interpreted languages. Java is compiled into an intermediate binary code called Java Virtual Machine (JVM) byte code. Then byte code is interpreted to run the program.

## 6.4. Common Programming Languages

There are a wide variety of programming languages, each created with specific goals in mind:

Language	"Hello world!" example	Typical Users
C	<pre>#import &lt;stdio.h&gt;  int main(void) {     printf("Hello world!");     return 0; }</pre>	embedded systems and operating systems Programmers.
C++	<pre>#import &lt;iostream&gt;  int main() {     std::cout &lt;&lt; "Hello world!"; }</pre>	Windows application and game developers
Python	<pre>print("Hello world!")</pre>	machine learning, Data Scientists
Swift	<pre>Swift.print("Hello world!");</pre>	Apple IOS application developers
Java	<pre>public class JavaProgram {     public static void main(String[] args) {         System.out.println("Hello world!");     } }</pre>	Android application developers

## 6.5. Software Development Steps

The initial tendency of a new software developer is to start typing code as soon as possible. For most real problems, this approach is likely to take longer, and result in lower value solutions that have to be reworked. Typically, the end result is hard to understand and maintain.

The earlier steps in software development process have a much higher impact on the effectiveness and quality of results than later tasks such as the coding. All successful businesses have discovered this fact and have their most skilled developers work on the initial phases of development, and less experienced developers work on the coding and testing.

Although the software development steps vary among organizations, depending upon their needs and their software development maturity, most successful software development processes can be organized into four steps:

### Requirement Analysis

The requirement analysis is the most important step in the programming process. This is where the programming problem is defined. The more details that can be agreed to early on, the higher the probability of project success.

It is important for software developers to spend sufficient time and effort to fully understand the problem and customer needs. Additionally, the following points should be documented and agreed upon in this phase:

- Know the customer
- Itemize a list of functions that the program has to perform
- Reliability requirements
- Response requirements
- Usability and user interface requirements
- Estimated time to complete the design and implement the solutions
- List of unresolved issues and a plan to resolve them.

### Design Phase

There are many design processes and tools available on the market to document high-level design before moving on to the coding. In this section, we discuss the use of pseudo code and flow charts as two common design tools.

#### Pseudo code

The most common method used to document software design is pseudo code. Pseudo code is a mix of English and programming language elements (with limited regard for correctness of syntax). The goal is to convey the architecture, major functional blocks and interfaces of your design to the reader and implementer. It also helps the designer to concentrate on the algorithms without the syntax details. Note that the pseudo code cannot be compiled or executed.

Although we will not introduce specific rules for pseudo code (no specific rules for pseudo code, write it however you can understand it), the designer should attempt to convey sufficient information so that the implementer can use the pseudo code to implement the final syntactically correct code. There is a fine balance between enough detail and clarity of the conceptual design. Most larger organizations have software development guidelines that provide specific guidance for development process including pseudo code.

- Example: Pseudo code for a program that counts from 0 to 10 and returns.

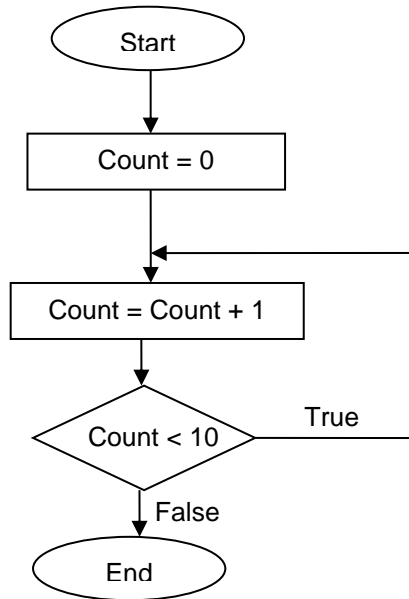
```
Count=0
```

```
Count up to 10
Return
```






### Flow Chart

Flow charts allow for more detailed specifications and program flow. They are also graphical instead of text-based so they are easier to read and understand for the typical programmer and reviewer than pseudo code. The drawback is that flow charting for a complex project can become impractical due to the paper/computer screen size limitations. Many projects use flow charts for areas that require detailed flow definition while using the pseudo code for the rest of the design. There are a number of tools on the market for flow charting.

**Example:** Write a program that counts from 0 to 10 and returns when done with the counting.



#### Basic Flow Chart Components

-  Oval holds start & end
-  Box holds actions
-  Diamond holds decisions (use True/False for results)
-  All connecting links must be arrowed lines (flow).
-  Input/output

### Coding

Finally, this is the step in which we actually start to type commands into our development environment or write the program. It is important to use the design definition as a guide for the coding. If a design or requirements error is found, it is important to update the earlier steps to correct the error before continuing with the coding task.

### Testing and Validation Step

As the name implies, the Testing and Validation step ensures that the final program/code meets all the requirements agreed to at the start of the development process. If the requirements have been done with care and sufficient effort, then the testing would simply involve testing the program versus the requirements. Again, it is clear that investment in requirements pays dividends throughout the process.

## **Chapter 7. Programming in Python**

### **7.1. Key Concepts and Overview**

- Getting Started with Python
- Python variables and Operators
- Creating and Running Python Scripts
- Python Flow Control
- Python Built-in Functions
- Python Modules
- Python User Defined Functions
- Python Quick Reference

## 7.2. Getting Started with Python

Python is a high-level programming language, designed for quick implementation with clear syntax and minimal restrictions. Therefore, Python is great first high level programming language for new programmers. Python is supported across a variety of environments including Linux, Windows, and IOS.

Python's extensive function libraries and the fact that it does not require compiling before execution, makes it an ideal programming tool for prototyping ideas and exploring new algorithms and concepts quickly. Python is a widely used language by beginners and professional alike. It is the language of choice for Machine Learning, Data Science and Big Data Analysis.

### Installing Python

There are multiple development environments available for Python programming. In this section, we will be using Python 3.7 but the instructions and processes included in this section should be compatible with other versions. The user interface elements in other versions may be different.

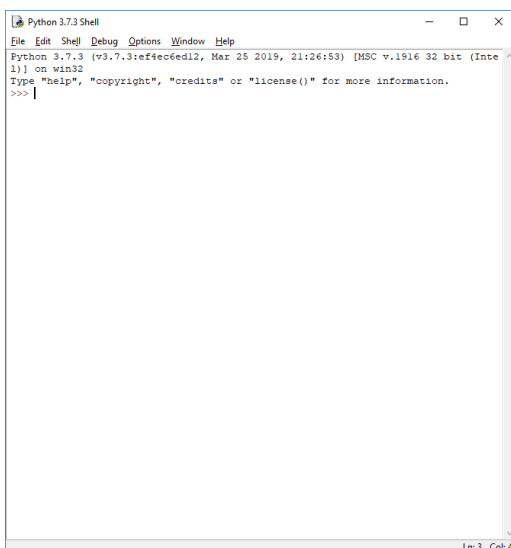
To download Python 3.7, visit the webpage <https://www.python.org/downloads/release/python-373/>. Download the release for your operating system. Once downloaded, run the executable to complete the installation process.

### Python Integrated Development Environment (IDLE)

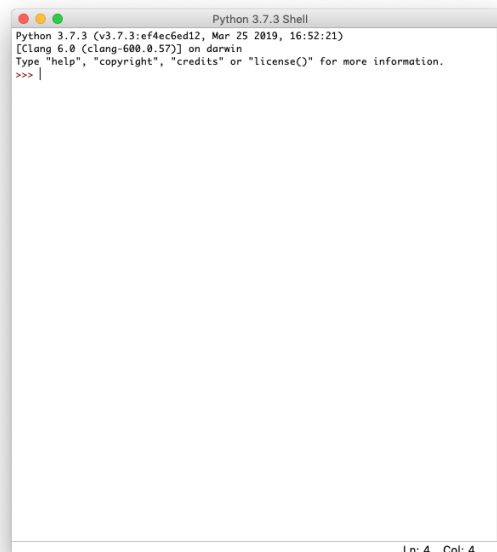
There is nothing inherently special about code files; they are simply text files that have instructions meant for a computer, rather than a human. As such, most code files could be opened and edited in a program like Notepad (on Windows) or TextEdit (on macOS). However, since these programs are not meant for editing code, they can be cumbersome to use.

As a result, programmers like to use what is called an Integrated Development Environment (IDE), IDEs provide a text editor that is designed with features and intelligence to help programmers develop software faster with less error. One of the most useful feature is that when typing a command, IDE shows the correct syntax and also color code various part of command for ease-of-programming or error detection. Python comes with an IDE called IDLE.

To start a session type in the computer's search bar "IDLE" and run it. You should see one of the following windows (IDEL shell Window):



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
Ln: 3 Col: 4
```



```
Python 3.7.3 Shell
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 16:52:21)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
Ln: 4 Col: 4
```

### Writing Your First Line of Python



It is a common tradition for those learning a new programming language to start by writing a program that displays or prints "Hello world!" on the screen. This can be done in Python quite easily. Type the following line into IDLE and press Enter/Return:

```
>>> print("Hello world!")
```

The IDLE window should immediately write out, on the line below your code:

```
Hello world!
```

Congratulations - you have just written your first Python program! As you might have guessed, this program takes whatever is within the quote marks ("" ) and prints it to the screen.

**Student Exercise** – First Program

Write a line that will write "Python is cool!" to the screen.

**Solution**

### 7.3. Python Variables and Operators

Python is capable of much more than just writing sentences to the screen. One of its most useful features is its ability to store and evaluate expressions.

#### Variables in Python

A variable is a name given to a memory location, where a value can be stored. The variable name should be selected so that it represents the content of the memory. For example, if you are counting apples, it is good idea to name it `apple_count`. Here are some constraints on naming variables:

- Must start with a character (A-Z, a-z or underscore “\_”)
- Cannot start with a number
- Cannot have a space commonly underscore “\_” is user in place of space
- Python is case sensitive so variable name `x` is not the same as `X`.

There is no need to declare a variable. A variable will be automatically created the first time it is used in your code (commonly called script in Python). Also, the variable type is determined based on the value being assigned.

#### Example - Variables

Create a variable `x` and set it to value 5

#### Solution

```
>>> x = 5
```

If you type this line in IDLE, and hit Enter/Return, variable `x` is automatically created and assigned integer value of 5.

If you type `x` and hit Enter/Return again, the value of `x` will be displayed:

```
>>> x
5
```

Numbers with decimal points, which are called floats, can be used just as easily:

```
>>> y = 3.14159
>>> y
3.14159
```

Numbers are not the only things that can be stored in variables. A string (series of characters within a single or double quotation marks), can also be stored in a variable.

```
>>> a = "Variables are neat"
>>> a
Variables are neat
```

*Note: Python accepts single ('), double (") and triple (""" or ''') quotes to denote string literals, as long as the same type of quote starts and ends the string. The triple quotes are used to span the string across multiple lines.*

#### Arithmetic Operators

Python has the ability to do arithmetic on variables and values in order to compute new values. Type the following line into IDLE and press Return/Enter.

```
>>> 2 + 2
```

The number 4 should be printed out in response which is the result computing . You can assign the value to a variable as shown below:

```
>>> a = 2 + 2
>>> print(a)
4
```

Addition is not the only arithmetic function Python provides. Here is a list of some of the common arithmetic operators Python provides:

Type	Syntax	Meaning	Example
Addition	$a + b$	$a + b$	>>> 2 + 3 5
Subtraction	$a - b$	$a - b$	>>> 10 - 2 8
Multiplication	$a * b$	$a \times b$	>>> 4 * 5 20
Division	$a / b$	$\frac{a}{b}$	>>> 16 / 4 4.0
Exponent	$a ** b$	$a^b$	>>> 2 ** 6 64
Modulus (Remainder)	$a \% b$	$a \bmod b$	>>> 10 \% 3 1

*Note: You may have noticed that the division operator returned a value with a decimal point. This has to do with the different Python data types of Python which will be discussed later.*

### Student Exercise - Arithmetic

Using Python and IDLE, compute the value of the following expression (3\*2+8/3)

### Solution

### String Arithmetic

In addition to performing arithmetic on numbers, Python performs operations on strings. The most common one is called the concatenation operator. When two strings are “added” together, the resultant string is the second string stuck to the end of first string as show in the following example:

```
>>> "Python is my " + "favorite language"
'Python is my favorite language'
```

*Note: If the computer printed “Python is myfavorite language”, you did not include the space after the word “my”. There is no implied space between strings when they are concatenated.*

A string can also be multiplied by a number to repeat it. Here is an example:

```
>>> "Knock " * 3
'Knock knock knock '
```

## Using Variables as Values

Variables can take the place of numbers in calculations like so:

```
>>> a = 3
>>> a + 2
5
```

On the second line, the Python interpreter replaces the variable `a` with the number 3, since that is the value stored in `a`.

Of course, this also works with strings:

```
>>> b = "CSE 120"
>>> "My favorite class is " + b
'My favorite class is CSE 120'
```

As a general rule, remember that wherever you can put a literal value (like 5 or “Hello world”), you can also put the name of a variable holding something of that same type!

## Order of Operation and Parentheses

Normally, Python will follow the traditional order of operations:

1. Exponents (highest priority – gets done first)
2. Multiplication & division, left to right
3. Addition & subtraction, left to right (lowest priority – gets done last)

Much like traditional mathematics, we can change order of operation by the use of parentheses. Here is an example of how changing order of execution effects the results:

```
>>> 2 * 5 + 1
11
>>> 2 * (5 + 1)
12
```

### Student Exercise A – Order of Operation

Use Python to evaluate the expression  $\frac{(1+3)^2}{5}$ .

### Solution

### Student Exercise B – Order of Operation

Find the solutions to the quadratic equation  $2x^2 - 14x + 24=0$

Hints:

1) roots of quadratic equation  $ax^2 + bx + c=0$  can be found using:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2)  $\sqrt{x} = x^{0.5}$ .

### Solution

### Boolean, Logical and Relational Operators

In addition to integers, floats, and strings, Boolean is another type of data type often used in Python. These variables can be only `True` or `False`. Setting a direct Boolean value in Python is, of course, quite simple:

```
pythonIsEasy = True
pizzaIsBad = False
```

### Logical Operators

Logical operators operate on Boolean values. “and”, “or”, “not” are three of the common logical operators in Python.

- `a and b` returns `True` only if both `a` and `b` are `True`, otherwise returns `False`
- `a or b` returns `False` if neither `a` or `b` is `True`, otherwise returns `True`
- `not a` returns `True` if operand `a` is `False` otherwise returns `False`

## Relational Operators

Relational Operators are used to compare two Boolean variables, variable or expressions:

Type	Syntax	Meaning	Example
Equal to	$a == b$	$a = b$	<pre>&gt;&gt;&gt; 5 == 5 True &gt;&gt;&gt; 3 == 5 False</pre>
Not equal to	$a != b$	$a \neq b$	<pre>&gt;&gt;&gt; 5 != 5 False &gt;&gt;&gt; 3 != 5 True</pre>
Less than	$a < b$	$a < b$	<pre>&gt;&gt;&gt; 6 &lt; 3 False &gt;&gt;&gt; 1 &lt; 3 True</pre>
Less than or equal to	$a \leq b$	$a \leq b$	<pre>&gt;&gt;&gt; 1 &lt;= 3 True &gt;&gt;&gt; 3 &lt;= 3 True</pre>
Greater than	$a > b$	$a > b$	<pre>&gt;&gt;&gt; 4 &gt; 2 True &gt;&gt;&gt; 2 &gt; 2 False</pre>
Greater than or equal to	$a \geq b$	$a \geq b$	<pre>&gt;&gt;&gt; 4 &gt;= 2 True &gt;&gt;&gt; 2 &gt;= 2 True</pre>

## 7.4. Creating Python Program Files (Script)

Up to now, we have been typing one line of code (interactively), evaluating it, and then entering the next line of code. It would be much more efficient to write all of the commands and have them all executed together. The file that contains all the commands is called a code file or more commonly a Python script.

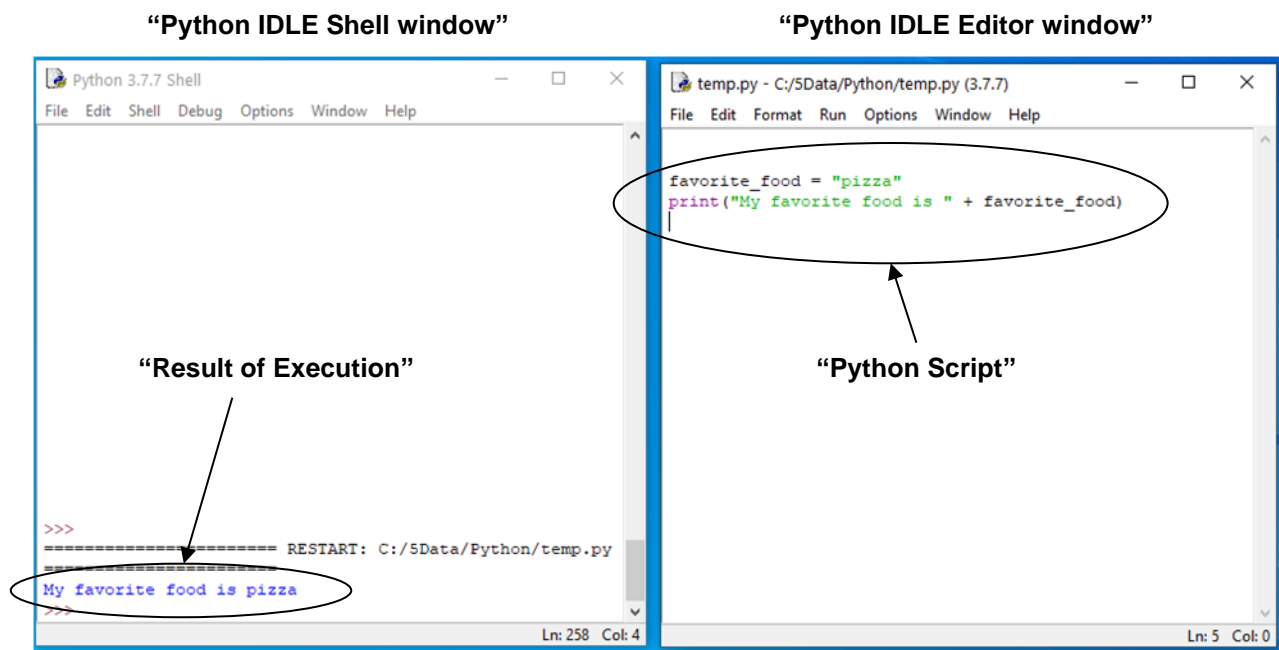
To create a python script, click the “File” then “New File” from the IDLE Shell window menu (File>New File). This opens a new Editor window that you can use to type in your Python code.

You can type multiple lines of code without having them instantly execute. Try typing in the following:

```
favorite_food = "pizza"  
print("My favorite food is " + favorite_food)
```

When done, use the Editor window File menu to save the file (File>Save As). Notice that the file has a “.py” extension which identifies it as an executable Python script.

To Run the script, in the Editor Window Run menu click the Run Module (Run> Run Module). This will execute your script and should see the result in the IDLE Shell window



### Comments

While Python is a high-level programming language and easy to read, it is helpful to add comments that are ignored by the computer, but human reader can benefit from. Your goal with commenting is not to explain the Python command or syntax since the human reader is expected to know Python program. The reason for commenting is to help human reader understand the functionality of the code without having to read and understand every line of code.

Any writing after hashtag symbol “#” to the end of line is ignored by the computer and is consider a comment. Here is an example of comment

```
# my simple program
print("Hello there!") # a friendly greeting
```

When the Python runs your script, it ignores all the comment, so it only sees:

```
print("Hello there!")
```

If there is too much information in your comment to fit in a single line, you can use three sets of single or double quotes ( ' ' ' or " " " ) to establish a multi-line comment. Anything within the quotes will be considered comment.

```
"""
A simple Hello World program
Created for ENGR 120
"""
print("Hello world! ")
```



## 7.5. Python Flow Control

Computer executes programs just like a human. It starts from the beginning of the file executes each code line until it reached the end. There are times when we want to execute a set of commands multiple times and skip other commands. We need to have a way to break this line-by-line execution. Python designer have seen this need and provided us with flow control statements such as loops, if-else and other constructs.

### for Loops “Collection Controlled Loops”

If we want to write a program that prints every number from 1 to 100, one way to accomplish this goal would be to write 100 `print()` statements in a row, like so:

```
print(1)
print(2)
print(3)
...
print(100)
```

While this would meet the requirement, it is tedious and not the best approach. Not to mention, next time you may need to print numbers from 1 to 1,000,000? In this case, the earlier approach would not be practical. Fortunately, Python has “For Loop” flow control statement which simplifies the task. Using the “For Loop” statement, we can reduce the 100 lines code to the 2 lines of code shown below:

```
for x in range(1,101):
    print(x)
print("got the job done in 2 lines!")
```

All the lines after “`for x in range(1,101):`” that is indented will run for  $x=1$  through  $x<101$ . In this case only `print(x)` is indented (tabbed or spaced in). Note that `print("got the job done in 2 lines!")` is not indented so it is not part of “For Loop”.

### Lines and Indentation

Python has no braces to indicate blocks of code for function definitions or flow control. Blocks of code are denoted by indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. Thus, in Python all the continuous lines indented with same number of spaces would form a block.

```
for x in range(4,20):
    print ("\ncount is = " , x)
    print ("\ncount square is = " , x*x)
print ("This line is not indented or spaced-in so is not in the loop")
```

In this example, the last “print” is not indented so it is not part of the loop and will only execute once after the loop is done.

### Student Exercise – for Loop

Write a Python script that prints the value of  $y=2x+1$  for  $x$  between 120 to 165 using the “For Loop” statement.

### Solution

### “for Loop” General Form

Earlier examples show how “For Loop” statement can be used to loop (iterate) through values of a variable in given range. “For Loop” is also able to loop through strings and other sets or collection.

Here is the general form of “For Loop”

```
for [item] in [iterator]:  
    [code to loop]
```

[item] – Variable to store each item as it iterates or loops through  
[iterator] – A set of collections to loop through, like a range() function or string  
[code to loop] – The code to be executed through the loop. Must be indented.

Here is an example of iterating (looping) through characters of a string and printing each character:

```
for a in "engrcs":  
    print(a)
```

The results are:

```
e  
n  
g  
r  
c  
s
```

### while Loops “Condition-Controlled Loops”

“While Loops” repeat the [code in the loop] until the [condition] is met.

**General Form:**

```
while [condition]:  
    [code to loop]
```

[condition] – Condition is a Boolean value or expression. As long as the condition is True, the code in the loop executes. Once condition is False, Python skips the code in the loop and continues with command after the loop.

[code to loop] – This code will execute every time through the loop.

### Example – While Loop

Write a Python script that prints every other number from 5 to 99

### Solution

```
Num = 5          # look a variable name with more than one character  
while (Num < 100):  
    print (Num)  
    Num = Num + 2
```

### if-else Statement “Conditional”

Conditional statement “If-else” enables a program to test a condition and based on the results (True or False) execute a different set of code. Here is a simplified general statement and the correspond flow charts:

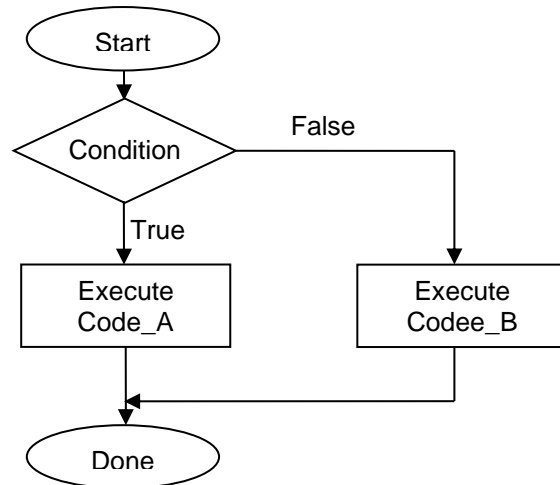
#### “if-else” Simplified General Form

```
if [condition]:  
    [code_A]  
else:  
    [code_B]
```

[condition] – If this Boolean expression [condition] is True then execute [code\_A] otherwise, execute [code\_B]

*Note: to reduce complexity “elif” option is not included in this section*

#### “if-else” Flow Chart



#### Example – “if-else” statement

Write a script to print message that states if my\_num is greater than 3 or it is not greater than 3.

#### Solution

```
# This code compares a with 3  
my_num = 5  
if my_num > 3:  
    print("my_num is greater than 3")  
else:  
    print("my_num is not greater than 3")  
# end of code segment
```

#### Example – “if-else” statement

Write a script to increase the variable count if count is less than limit otherwise count is decremented. Print the final value of count.

#### Solution

```
# Initialize Variables  
limit = 100  
count = 35  
if count < limit:          # compare with the limit  
    count = count + 1  
else:  
    count = count - 1  
print("count is equal to ", count)  
# end of code segment
```

## 7.6. Python Built-In Functions

Python has an extensive set of built-in functions, such as `print()` and `range()` that we have already used. Before writing your own function, it is always a good idea to carefully check the full list of Python built-in functions at the following link:

<https://docs.python.org/3/library/functions.html>

This section provides descriptions of a few more commonly used Python built-in functions such as `print()`, `input()` and data type converters.

### **print() Function**

prints a string based on the arguments passed. Below is general form for `print()`:

```
print (" [formatting string]" % ([arg1, arg2, ...]))
```

[formatting string] – a string that includes text and operator (`%x`) that specify where in the string arguments should be placed.

[arg1, arg2, ...] – arguments that will be placed in the text before printing .

### **Example – print()**

The following `print ()` function call:

```
print("My name is %s and I am %d years old. " % ("Billy", 21))
```

prints the following results:

```
My name is Billy and I am 21 years old.
```

As you can see `%s` was replaced by string "Billy" and `%d` was replaced by decimal integer 21. `%s` and `%d` are referred to as formatting Symbol. Here is sampling of other formatting symbol you can use:

Format Symbol	Expected argument or Function
<code>%s</code>	string
<code>%d</code>	signed decimal integer
<code>%f</code>	floating point numbers
<code>%x</code>	hexadecimal integer
<code>\t</code>	Insert a tab
<code>\n</code>	Inset a new line

### **Example – print ()**

The following `print ()` function call:

```
print("My name is %s" % ("Billy"))  
print ("\n\n")  
print ("My age in decimal is %d \tand\t My age in hexadecimal is %x" %  
(21,21))
```

prints the following results:

```
My name is Billy
```

My age in decimal is 21                      and                      My age in hexadecimal is 15

### **input() Function**

Now that we can print information for our program users to see, it would be helpful to be able to ask users questions and get their response. `input()` provides this capability. We can use `input()` to ask questions and get response from the user. Here is a general form for `input()`:

```
response = input ("display text")
```

[display text] – Python prints the [display text] on the screen and waits until user enter a response and presses <Enter or Return> key.

[response] – Python saves the user response in [response]

### **Example – input()**

The following code:

```
text = input("enter something: ")  
print(text)
```

This code reprints user's input text unmodified.

**Student Exercise** - Try saving and running the above code program. When it starts, type some text and then hit Enter/Return. What happens?

### **Solution**

### **Example – input()**

Write a program that asks for the user's name. Once it has been entered, the program should print "Ah, so your name is" and add user's name.

### **Solution**

```
text = input("what is your name? ")  
print("Ah so your name is " + text )    # using + to concatenate or merge strings
```

**Student Exercise** - Try saving and running the above code program. When it starts, type your name and then hit Enter/Return.

### **Solution**

## Data Type Conversion

Python automatically defines user input as a string so if you are not expecting string then your results may not be correct.

Let use the following code to demonstrate the idea:

```
a = input("Enter number a: ")
b = input("Enter number b: ")
print(a + b)
```

Case 1) If user inputs "Fine " for a and "Day" for b, this code display "Fine Day" which is concatenation of the two strings.

Case 2) If user inputs 5 for a and 6 for b, this code displays 56 instead of 11 since it assumes that a and b are strings . So if you wanted them to be treated as an integer and added then you need to change the code by casting (converting) the input to integers as shown below:

```
a = int(input("Enter number a: "))
b = int(input("Enter number b: "))
print(a + b)
```

If we want to be able to use input() and be sure that they are typed correctly, then you need to use the casting (conversion) functions to force the variable to your desired type. Here are three of the most common casting functions::

- **int(value)** converts a given value to an **integer** (whole number)
- **float(value)** converts a given value to a **floating-point number** (number with a decimal point)
- **str(value)** converts a given value to a **string**

## len() "string length"

len function accepts a string as an argument and returns the number of characters in the argument.

Here are a couple of examples:

```
print(len("Bye")) # prints 3
print(len("Hello world")) # prints 11
```

## help() "Getting Help"

help() accepts name of a function/object as the argument and returns description of the function/object and how to use it.

Typing help("len") in the IDLE Shell Window returns:

*Help on built-in function len in module built-ins:*

*len(obj, /)*  
*Return the number of items in a container.*

## 7.7. Python Modules

A module in Python is a file consisting of existing Python code. Modules are used to logically organize Python code by grouping related code into a module. Python includes a rich set of built-in modules. To find out what modules python has, run the following command from IDLE Shell window:

```
>>> help("modules")
```

You should see a long list of module names. This section uses a couple of commonly used modules to demonstrate the usage of modules.

### Importing Modules

The first step is to import the module you need before using the functions from the module. You can use the following to import the code:

```
import <module_name>
```

Where `module_name` is the name of the module you want to import.

You can import multiple modules with a single `import` statement as show below:

```
import module_1, module_2,..., module_n
```

In the remainder of this section, `math` and `time` modules are covered.

### math module

`math` module provides more complex math operation than the basic operations discussed earlier in this chapter. The following table shows some of the functions in `math` module:

Function	Returns
<code>math.exp</code> ( $x$ <float, int>)	$e^x$
<code>math.log</code> ( $x$ <float, int>, $b$ <float, int>)	If $b$ is specified, returns $\log_b x$ . If $b$ is not specified, returns $\ln x$ .
<code>math.sqrt</code> ( $x$ <float, int>)	$\sqrt{x}$ .
<code>math.sin</code> ( $x$ <float, int>)	$\sin x$ where $x$ is in radians.
<code>math.cos</code> ( $x$ <float, int>)	$\cos x$ where $x$ is in radians.
<code>math.tan</code> ( $x$ <float, int>)	$\tan x$ where $x$ is in radians.
<code>math.e</code>	value of $e$ , approximately 2.7183.
<code>math.pi</code>	value of $\pi$ , approximately 3.1415.

*Note:* More information about these and other functions can be found at <https://docs.python.org/3/library/math.html>.

### Example – math module

Write a program where the user can type in a value and get the sine of that value using the `math` module.

### Solution

```
import math          # import the math module
# Get the user input and convert it to a float
in_val = float(input("Enter a angle value in radians: "))
# Get the sine of the user inputted value
sine_of_val = math.sin(in_val)
```

```
# Print out the result
print("The sine of %f is %f" % (in_val,sine_of_val))
```

### Student Exercise – math module

Write a script that accept angle in radians from user and prints the sine, cosine, and tangent of the angle.

Expected usage/output:

```
Enter a number to get the sine, cosine, and tangent of!
> 0.5
The sine of 0.5 is 0.4794
The cosine of 0.5 is 0.8776
The tangent of 0.5 is 0.5463
```

### Solution

#### time module

The time module contains useful functions for dealing with time. If you plan to use this module, remember to import it:

```
import time
```

The following table shows some of the functions in time module:

Function	Returns
time.localtime()  time.localtime().tm_year time.localtime().tm_mon time.localtime().tm_mday time.localtime().tm_hour time.localtime().tm_min time.localtime().tm_sec time.localtime().tm_wday time.localtime().tm_yday time.localtime().tm_isdst time.localtime().tm_zone time.localtime().tm_gmtoff	struct_time stores elements of time and date  current year current month current day of the month current hour of the day current minute of the hour Current second of the minute an integer representing the day of the week (0 is Mon and 6 is Sun.) the day in the year 1 for daylight saving, 0 when not, -1 for unknown a string for time zone; for example "PDT" for Pacific Time. number of seconds the time zone is off of UTC
time.sleep(x <float, int>)	Pauses program execution for x seconds.
time.gmtime(secs <float, int>)	Creates a struct_time object for secs seconds since the epoch (often number of seconds since January 1, 1970).

Note: More information about these and other functions can be found at <https://docs.python.org/3/library/time.html>.



### Example - time module

Write a program that uses the `time` module to print out the current day of the year (for example, January 1 would be 1).

### Solution

```
import time          # import the time module
print(time.localtime().tm_yday)
```

### Student Exercise – time module

Write a program using the `time` module that asks the user to input a number of seconds to wait, then ends the program. (Use the function `time.sleep()`).

### Solution

### **sys.exit() “Python script terminating itself”**

The `sys` module contains many advanced system-level functions that can be executed in a Python script. The one that could be useful for beginners is `sys.exit()`. This function terminates the Python Script upon execution.

### Example – sys.exit()

```
# make sure to import sys module first!
import sys

print("This should show up")
sys.exit() # this will stop execution and terminates the script
print("But this should not")
```

## 7.8. Python User Defined Functions

We have already used a function, `print()`, in earlier sections. When you type `print("Hello world")`, the `print()` function is called, and the string "Hello world" is sent in as an argument. Functions in programming languages are self-contained “mini-programs” that are dedicated to a specific task. Functions allow a large program to be divided in smaller programs or functions. The main benefit of function is that you write it once and you can call it anytime you need to perform the task. Functions enable reuse and modularization.

### Defining & Calling Functions

While `print()` is built into Python that we use (call), We can also create or define our own functions. So for a new function that is not built in (provided in existing libraries), first step is to define the new function before the new function can be called (used).

#### Example – Defining a Function

Write a new function that accepts three numbers and print their sum.

#### Solution

The following code defines the function `print_sum_of_numbers`

```
def print_sum_of_numbers(num_a, num_b, num_c):
    result = num_a + num_b + num_c
    print(result)
```

To call (or use) this function:

```
y = print_sum_of_numbers(3,2,5)      # after the call, y = 3+2+5 = 10
```

### Function Definition and Call General Form

First step is to define a function:

```
def [function_name]([arg1, arg2, ...]):
    [code to run]
    return(value)
```

[function\_name] – The name of the function

[arg1, arg2, ...] – arguments or variables you want to pass in to be used in the function

[code to run] – Code to be executed when function is called

return (value) – use this statement at the end of the function if you would like to return a value

Second step is calling (using) the function:

- If function code has “return (value)” line at the end of function definition, here is the call form:  
`results = [function_name]([arg1, arg2, ...])`
- If you do not have a “return (value)” line at the end of your function definition, here is the call form:  
`[function_name]([arg1, arg2, ...])`

The definition must be in the code file before the call since Python is an interpreted language.

#### Example – Defining and Calling Functions with return

Define a function that returns results of evaluating expression  $(5x^2 + 20y^3 - 100)$  and accepts  $x$  and  $y$  as arguments. Next, Call the function with  $x=6$  and  $y= - 3$  and print the results.

### Solution

```
# Defining a function that returns a value
def eval_expression(x, y):
    value = 5*(x**2) + 20*(y**3) - 100
    return value

# Calling the function that return value with x=6 and y=-3
num2print = eval_expression (6, -3)
print("results of calling the function = ",num2print)
```

### Example – Defining and Calling Functions without return

Define a function that evaluates expression  $(5x^2 + 20y^3 - 100)$ , accepts  $x$  and  $y$  as arguments and print the results for  $x=6$  and  $y= - 3$ .

### Solution

```
# defining a function that returns a value
def eval_expression(x, y):
    value = 5*(x**2) + 20*(y**3) - 100
    print("results of calling the function = ",value)

# calling the function with x=6 and y=-3
eval_expression (6, -3)
```

### Student Exercise – Function definition

Write a function that multiplies two input numbers and prints the results.

Example call or usage:

```
multiply_two_numbers(6, 7)
```

Example output:

```
49
```

### Solutions

## 7.9. Python Quick Reference

### Data Types

`string` – Used for storing letters and text.

Examples: "a" , "Hello world", "I go to school at Clark College."

`int` – Used for storing whole numbers.

Examples: 0, 52, -13

`float` – Used for storing fractional numbers.

Examples: 0.0, 2.3, -32.4

`bool` – Used for storing true/false values.

Examples: True, False

### Comments

Single-line comments – Use a hash tag symbol (#) to comment out everything on the line that follows

```
# this is a comment
print("Printing to the console") # another comment
```

### Arithmetic Operators

Type	Syntax	Meaning	Example
Addition	$a + b$	$a + b$	>>> 2 + 3 5
Subtraction	$a - b$	$a - b$	>>> 10 - 2 8
Multiplication	$a * b$	$a \times b$	>>> 4 * 5 20
Division	$a / b$	$\frac{a}{b}$	>>> 16 / 4 4.0
Exponent	$a ** b$	$a^b$	>>> 2 ** 6 64
Modulus (Remainder)	$a \% b$	$a \bmod b$	>>> 10 \% 3 1

### Logical Operators

- `a and b` returns True only if both a and b are True, otherwise returns False
- `a or b` returns False if neither a or b is True, otherwise returns True
- `not a` returns True if operand a is False otherwise

## Relational Operators

Type	Syntax	Meaning	Example
Equal to	<code>a == b</code>	$a = b$	<pre>&gt;&gt;&gt; 5 == 5 True &gt;&gt;&gt; 3 == 5 False</pre>
Not equal to	<code>a != b</code>	$a \neq b$	<pre>&gt;&gt;&gt; 5 != 5 False &gt;&gt;&gt; 3 != 5 True</pre>
Less than	<code>a &lt; b</code>	$a < b$	<pre>&gt;&gt;&gt; 6 &lt; 3 False &gt;&gt;&gt; 1 &lt; 3 True</pre>
Less than or equal to	<code>a &lt;= b</code>	$a \leq b$	<pre>&gt;&gt;&gt; 1 &lt;= 3 True &gt;&gt;&gt; 3 &lt;= 3 True</pre>
Greater than	<code>a &gt; b</code>	$a > b$	<pre>&gt;&gt;&gt; 4 &gt; 2 True &gt;&gt;&gt; 2 &gt; 2 False</pre>
Greater than or equal to	<code>a &gt;= b</code>	$a \geq b$	<pre>&gt;&gt;&gt; 4 &gt;= 2 True &gt;&gt;&gt; 2 &gt;= 2 True</pre>

## If-else statement

```
if [condition]:
    [code_A]
else:
    [code_B]
```

If this expression [condition] evaluates to True then run [code\_A], otherwise execute [code\_B].

## For Loop

```
for [item] in [iterator]:
    [code to loop]
```

[item] – Variable to store each item of the iterator in

[iterator] – Something to loop over, like a range() function or string

[code to loop] – This code will be run as many times as [iterator] has items. While in this code, use the variable [item] to get the current item from [iterator].

## While Loop

```
while [condition]:
    [code to loop]
```

[condition] – This condition is checked every time before the code is run. If it evaluates to True, the code will be run; if it evaluates to False, it will be skipped.

[code to loop] – This code will continue to run until [condition] evaluates to False.

## Defining Function

```
def [function_name]([arg1, arg2, ...]):  
    [code to run]
```

[function\_name] – The name of the function

[arg1, arg2, ...] – Put arguments (variables you want to pass in to this function) here.

[code to run] – Code inside this block will be run when the function is called.

## Output

```
print([value])
```

Writes [value] to the console.

## Input

```
[response] = input (("[display text"]))
```

[display text] – Python prints the [display text] on the screen and waits until user enter a response and presses <Enter or Return> key.

[response] – Python saves the user response in [response]

## Casting/Type Conversion

```
# Convert [value] to an int (whole number) and saves it in i_value.
```

```
[i_value] = int([value])
```

```
# Convert [value] to a float (floating-point number) and saves it in f_value
```

```
[f_value] = float([value])
```

```
Convert [value] to a string and save it in s_value
```

```
[S_vlaue] = str([value])
```

## 7.10. Further Reading

- Python 3.7 Reference - <https://docs.python.org/3/>
- Tutorial and sample codes online

## 7.11. Problems

*"Use IDLE to write programs and include comments and test case for each assignment"*

---

1. In Python Shell use print function to display the string "Good morning!".
2. In Python Shell evaluate the expression,  $x = (3 * 2 + 20/5)$ , and use print function to display x.
3. Write a program to take Celsius degree and converts it to Fahrenheit. Print out the Fahrenheit values as integers.

*Hint:*

- 1) Conversion Formula  $\rightarrow f = 9/5*c + 32$
- 2) Use input function to accept user input

- 
4. Write a function that takes radius r, and calculates and returns the volume of a sphere for  $r > 0$ ; return 0 otherwise.  $V = \frac{4}{3}\pi r^3$

*Hint:  $\pi = 3.14$*

- 
5. Write a program that asks the user to enter a password. If they type the word "penguin", the program should print "Access Granted". If they type anything else, the program should print "Access Denied."
  6. Write a program to display all odd numbers from a range that is given by the user using input(). For example, if the user gives (5,11), the expected output is: 5, 7, 9, 11. Note: range start and end are inclusive.
  7. Write a program to print the lyrics to the song "99 Bottles of Beer". The output should begin with the following two verses:

*99 bottles of beer on the wall, 99 bottles of beer.  
Take one down, pass it around, 98 bottles of beer on the wall.*

*98 bottles of beer on the wall, 98 bottles of beer.  
Take one down, pass it around, 97 bottles of beer on the wall.*

This should continue, going down by one bottle, until the number of bottles reaches zero, at which point the following verse should be printed:

*No more bottles of beer on the wall, no more bottles of beer.  
Go to the store and buy some more, 99 bottles of beer on the wall.*

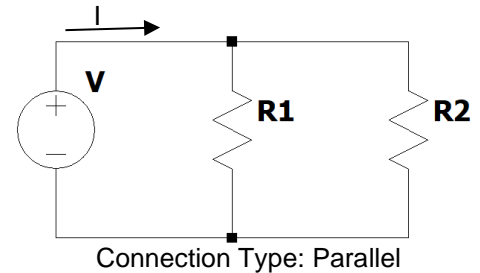
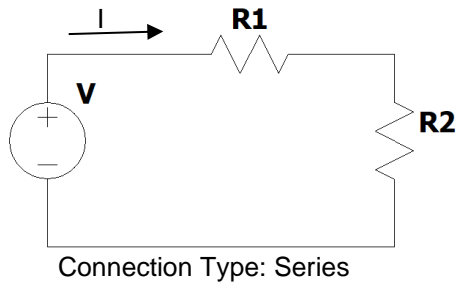
Note: Using what has been taught in this chapter, this can be done in only 7 lines of code!

Source: <http://www.99-bottles-of-beer.net/lyrics.html>

- 
8. Write a program that accepts a dollar amount 0.00 and 0.99 as an input and display exact changes for this amount using the minimum number of coins. For example, \$0.97 shows 3 quarters, 2 dimes, and 2 pennies.



- 
9. Write a program that requires four user input values:  $R_1$  and  $R_2$  (resistors),  $V$  (voltage) and connection type (Series or Parallel). The program calculates current ( $I$ ) from the voltage source and prints the value.



- 
10. Write a program that take a user input word and convert it to Unicode equivalent in hex and binary.

*Hints:*

1) Use *help()* for usage of built-in function *ord()*, *hex()* and *bin()*.

2) *Input/output example:*

*Enter a word: abc*

*Expected Output:*

*Word is: abc*

*In hex: 0x61 0x62 0x63*

*In binary: 0b1100001 0b1100010 0b1100011*

- 
11. Write a program that prints out different greeting strings based on the time of day from Python time module:

- From 00:00 – 11:59 prints "Good Morning!"
- From 12:00 – 16:59 prints "Good Afternoon!"
- From 17:00 – 19:59 prints "Good Evening!"
- From 20:00 – 23:59 prints "Good Night!"

*The program must also allow for manual entry of time to test the functionality.*

## **Chapter 8. Programming in MATLAB**

### **8.1. Key Concepts and Overview**

- Development Environment Interface and Structure
- Using the MATLAB Command Window
- Creating and Editing MATLAB M-files
- MATLAB Arithmetic and Logic Operators
- MATLAB Data Flow Controls

## 8.2. Development Environment Interface and Structure

We will be using **MATLAB** as the Interactive Development Environment (IDE) and programming language in this text. MATLAB is an example of a high level language and is well-suited to as an introductory programming language. MATLAB is the most common language used in engineering and science education, as well as in the industry to build models and solve engineering/scientific problems.

Today's engineering **requires models to test concepts before committing to production** or even development of prototypes. Science uses models to test theory and new concepts. Additionally, MATLAB's C-like programming environment provides an excellent preparation for moving to advanced programming tools.

### **MATLAB Overview**

MATLAB user interface changes regularly but the main components of user interface stays the same so use this section as introduction to the MATLAB environment and expect that the user interface for your version of MATLAB will be different.

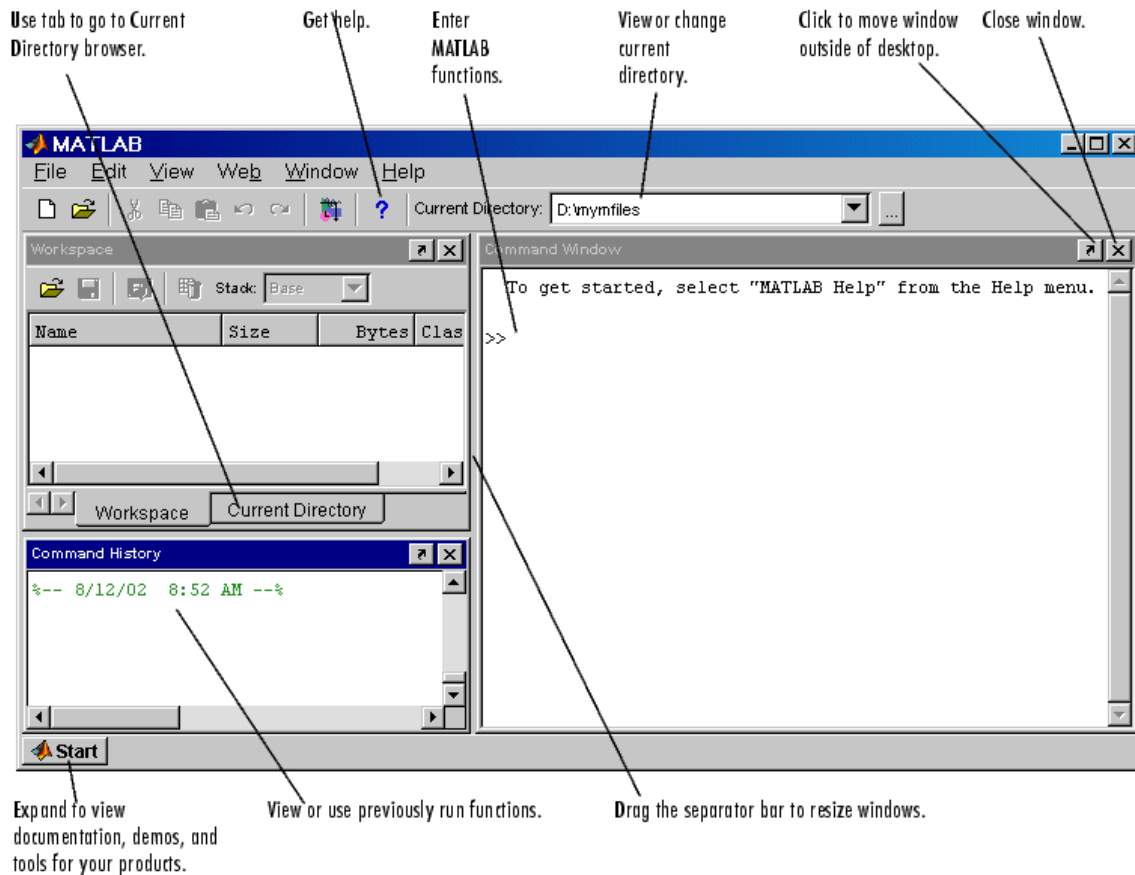
MATLAB system consists of five main parts:

- 1) **MATLAB Language**  
A high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.
- 2) **Mathematical Function Library**  
A collection of computational algorithms ranging from functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.
- 3) **Graphics**  
MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.
- 4) **Interactive Development Environment (IDE)**  
The IDE includes the MATLAB desktop and command window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.
- 5) **Application Program Interface (API)**  
This library enables programs written in C and FORTRAN to interact with MATLAB.

To start MATLAB, type MATLAB in the Windows search bar.

## Components of MATLAB Development Environment

The MATLAB IDE start up window looks like this:



**Each sub-window, or tab, can be rearranged to the user's liking**, but it is recommended to stay with the default configuration until the user is familiar with MATLAB.

## Editor/Debugger Window

In the MATLAB Development Environment, right click in the Workspace window to create a new test m-file, named `test.m`. **(Note that before a program can execute or run, it has to be typed in a file with the “.m” extension.)**

Once you have created a file, **double-clicking it causes MATLAB to open the file in the Editor/Debugger window.** The Editor/Debugger window is where programs are executed and results can be viewed.

Comment selected lines and specify indenting style using the **Text** menu. Find and replace strings.

Set breakpoints where you want execution to pause so you can examine variables.

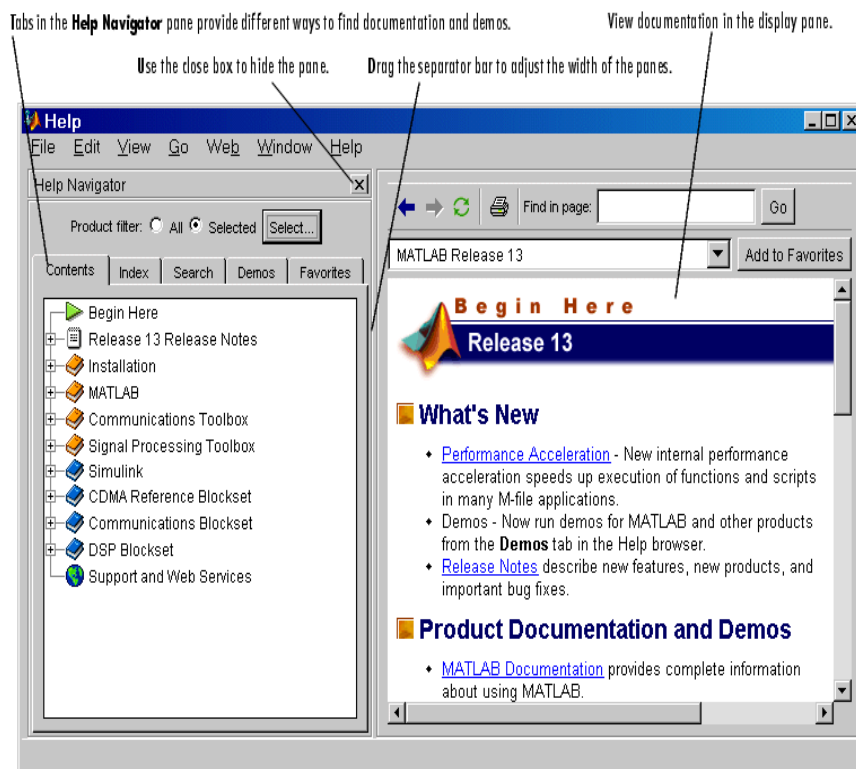
Hold the cursor over a variable and its current value appears (known as a datatip).

```
1 function sequence=collatz(n)
2 % Collatz problem. Generate a sequence of integers resolving to 1
3 % For any positive integer, n:
4 % Divide n by 2 if n is even
5 % Multiply n by 3 and add 1 if n is odd
6 % Repeat for the result
7 % Continue until the result is 1
8
9 sequence = n;
10 next_value = n;
11 while next_value > 1
12     if rem(next_value,2)==0
13         next_value = next_value/2;
14     else
15         next_value = 3*next_value+1;
16     end
17     sequence = [sequence, next_value];
18 end
```

collatz Ln 11 Col 16

## Help Section

From the MATLAB Development Environment, select *Help* → *MATLAB Help* to see the following window.



**MATLAB Help is the best resource available to learn about the full functionality of MATLAB.** The Tutorial Section can guide you through learning components of MATLAB, step-by-step. If you want information regarding a specific function, you can also **use the Search feature.**

**The demos in the Help Section are very helpful when learning MATLAB.** From the *MATLAB Help* window, select the *Demos* tab and review the various MATLAB demos to familiarize yourself with MATLAB's syntax and flow.

It is important to **become comfortable with the use of online documents to answer questions** and find any required MATLAB information.

### 8.3. Using the MATLAB Command Window

The command window allows you to write equations and issue commands that will be executed by MATLAB.

Each time you complete a command or when you start, `>>` indicates the location that the new command should be typed followed by the *Enter* key.

#### ❖ Clear Screen

The `clc` command **clears the command screen**. In the Command window, type `clc` and press *Enter*. Type the command and try it!

```
>> clc
```

**Note:** The most effective approach to learning MATLAB and programming in general is by performing the tasks while you are reading about them.

#### ❖ Help Command

The `help` command is one of the most useful commands since it **provides a brief description of any of MATLAB's available commands**. Simply type `help` followed by the name of the command you wish to learn more about. For example, the following command will display a description of the `clc` command with links to any related commands and more reference material.

```
>> help clc
```

**Use the *Help* menu items for complete descriptions of commands and their usages** since the `help` command only provides a brief description as a refresher.

#### ❖ Calculation & Semicolons

The command window **can also be used to solve equations and do arithmetic**, much like a calculator. For example, enter the following commands to make a calculation and save the results in `x`:

```
>> x = 10 + 12 * 20  
250
```

**By adding a semicolon ( ; ) at the end of command line before pressing *Enter* as shown below will cause the results of the calculation, to not be displayed:**

```
>> x = 10 + 12 * 20;
```

The semicolon is useful in programming **when there is a long list of questions, but only the final answer is of interest**. By adding them to the end of intermediate equations, **only the final answer will be displayed**.

#### ❖ Case-Sensitive Variables

**MATLAB is case-sensitive**, which means if you assign a value to `x` (lowercase), it will not be assigned to `X` (uppercase). To demonstrate this, try entering the following commands:

```
>> x = 10 + 5;  
>> x  
15
```

At this point, the command window will correctly display the value as 15. But now if you enter the following command:

```
>> X
??? Undefined function or variable 'X'.
```

The system will display the message “??? Undefined function or variable 'X'”. Although `x` (lowercase) was equal to 15, no value was assigned to `X` (uppercase).

The command window allows you to enter individual commands, but **it does not provide you with the ability to save multiple commands or execute multiple ones at once**. The ability to save and rerun a set of commands (program) and share it with others is critical for programs with more than a few commands. Fortunately, MATLAB offers such a capability through **M-files**.



## 8.4. Creating and Editing M-files

M-files allow the developer to write programs and save them for future use or later modification. It also opens the door to using functionality of a previously developed program in a new function.

### ❖ Creating an M-file

This process creates a **file with the extension “.m”, often referred to as an *m-file***. We can add a series of commands in the m-file separated by line breaks. The commands in the m-file can then be executed from the command window. Start by selecting the following from the command window’s menu bar:

*File > New > M-File* or *File > New > Function*

This will open a new window, the **M-file Editor, which can be used to enter commands**. Before we enter the desired commands, there are some lines that must be added to the beginning and the end of the file. First, add the line `function [results] = my_add (operand1, operand2)` to the beginning of the file, and on the following line type `end`. **Commands can now be typed in between these two lines, and they will be executed when the M-file is run.**

```
function [results] = my_add (operand1, operand2)
    results = operand1 + operand2;
end
```

The way we have defined the above function, it will accept two parameters, `operand1` and `operand2`, adds them together, and places the results in the variable `results`.

**In order to save the M-file, select *File > Save As***. At this point, you can browse and select the specific directory you want your file saved in. **It is required by MATLAB to name the function the same as the file**; in this case, the file must be named `my_add.m`. Take care to remember the file name and the saved directory, because you will need that information to run your MATLAB program.

Congratulations, you have created a MATLAB program!

**To run the program, open the command window and type the name of the file without the “.m” extension, but include the inputs/arguments**. For example, if you would like to add the two numbers 6 and 10, type the following in the command window:

```
>> my_add(6, 10)
```

MATLAB displays `ans = 16`, which is the result of adding 6 and 10.

### Notes:

- 1) **MATLAB’s case-sensitivity extends to file and function names**, so `My_add` is not the same as `my_add`.
- 2) **MATLAB does not allow the use of reserved words for variables, function names, or filenames. Reserved words are words that hold special meaning in the MATLAB language, like `function` or `end`**. The best practice is to customize the variables and names with the project specific designation to avoid potential conflicts.
- 3) The command window’s current directory must be the same as where the program is saved in order to execute it.

### Student Exercise – MATLAB Arithmetic

Write a program that accepts  $x$  as input and returns  $y = 2x + 30$ .

## Solution:

### Comments and Formatting Programs

It is important that programs contain information that helps any readers understand the function of the program, its current status, and its author(s). It is also useful to add comments describing complex functionality and assumptions that may not be obvious to the reviewer.

Percentage signs (%) inform MATLAB that the remainder of the line is meant as a comment for the human reader and is not to be executed.

Here is a properly commented program:

```
% m-file creation example
% Author: Izad Khormaei
% Last update: 9/10/06, Version: 1.2
% Input parameters: operand1, operand 2
% output parameter: results
% my_add function adds the value of operand 1 and operand2;
% assigns it to the
% results variable.

function [results] = my_add (operand1, operand2)
    results = operand1 + operand2;
end
```

Additionally, it is important to use indentation, such as tabs, to show the statement blocks for ease of readability. For example, the statement `results = operand1 + operand2` is indented to show that it is part of the function `my_add`.

If the editor window is not open, go to *File > Open* to open the m-file `my_add.m` and add comments to the file describing its function and author. Be sure to save the file before attempting to execute it from the command window.

**Student Exercise** - Write a program that calculates the volume of a sphere with radius  $R$  provided by the user.

$$\text{Hint: } V = \frac{4\pi}{3} R^3$$

### Solution

## 8.5. MATLAB Arithmetic and Logic Operators

MATLAB supports arithmetic, algebra, trigonometry and other mathematical operations for real numbers, complex numbers, and matrices. We will discuss the complex numbers and matrices later. For now, it suffices to consider them as a more generalized form of real numbers.

MATLAB supports a variety of operators, and the best source for learning about them is the *Help* section of MATLAB. Here are some definitions extracted from the MATLAB *Help* section for the most commonly used operators:

### Arithmetic Operators

Arithmetic operators detects the type of value being operated on and will perform the appropriate operation for the operator type.

- **Addition +**  
 $C = A + B \rightarrow$  adds A and B, with the result being stored in C.
- **Subtraction -**  
 $C = A - B \rightarrow$  subtracts B from A, with the result being stored in C.
- **Multiplication \***  
 $C = A * B \rightarrow$  multiplies A and B, with the result being stored in C.
- **Division /**  
 $C = A / B \rightarrow$  divides A by B, with the result being stored in C.
- **Power ^**  
 $C = A ^ B \rightarrow$  A is raised to the power of B, with the result being stored in C.

### Logical Operators

The logical operators **AND**, **OR**, and **NOT** are represented by the symbols **&**, **|**, and **~**, respectively. The value of the results and operands may be logical 0, representing false, or logical 1 (may also be any nonzero value), representing true. The logical operators return 1 (true) or 0 (false), as appropriate.

Operator orders of operation are:

NOT (~) evaluated first,  
AND (&)  
OR (|) evaluated last.

- $C = \sim A$  is defined by the following truth table:

❖ A	❖ C
❖ 0	❖ 1
❖ 1	❖ 0

- **AND &**  
 $C = A \& B$  is defined by the following truth table:

❖ A	❖ B	❖ C
-----	-----	-----

❖ 0	❖ 0	❖ 0
❖ 0	❖ 1	❖ 0
❖ 1	❖ 0	❖ 0
❖ 1	❖ 1	❖ 1

- **OR |**

$C = A | B$  is defined by the following truth table:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

### Relational Operators

Relational Operators enable developers to compare two operands and take action based on the relative value of the two operands. Note that when the relationship is true, the result will be logical 1 (true); otherwise the result will be logical 0 (false).

Here are a selection of available relational operators:

- **Less than <**  
( $A < B$ ) → True if A is less than B; otherwise False.
- **Greater than >**  
( $A > B$ ) → True if A is greater than B, otherwise False.
- **Less than or equal <=**  
( $A <= B$ ) → True if A is less than or equal to B; otherwise False.
- **Greater than or equal >=**  
( $A >= B$ ) → True if A is greater than or equal to B; otherwise False.
- **Equal ==**  
( $A == B$ ) → True if A is equal to B; otherwise False.
- **Not equal ~=**  
( $A ~= B$ ) → True if A is not equal to B; otherwise False.

### Examples – Relational Operators

Evaluate the value of C when  $C = \sim 5 \& 4 | 0$

#### Solution:

$C = (0 \& 1) | 0 = 0$  or False

### Examples – Relational Operators

Evaluate the value of C when  $C = A | B \& \sim A$  when  $A = \text{False}$  and  $B = \text{True}$

#### Solution

`C = 0 | (1 & 1) = 1 or True`

**Examples – Relational Operators**

Evaluate `{C == (A >= B) & (B <= C)}` when `A = C/B`, `B = 2`, and `C = 5.5`

**Solution**

```
{C == (2.75 >= 2) & (2 <= 5.5)}  
{5.5 == 1 & 1}  
{1}
```

**Student Exercise –Relational Operators**

Write a program that accepts two inputs, `in1` and `in2`, and returns 1 if the two values are the same.

**Solution**

## 8.6. MATLAB Data Flow Controls

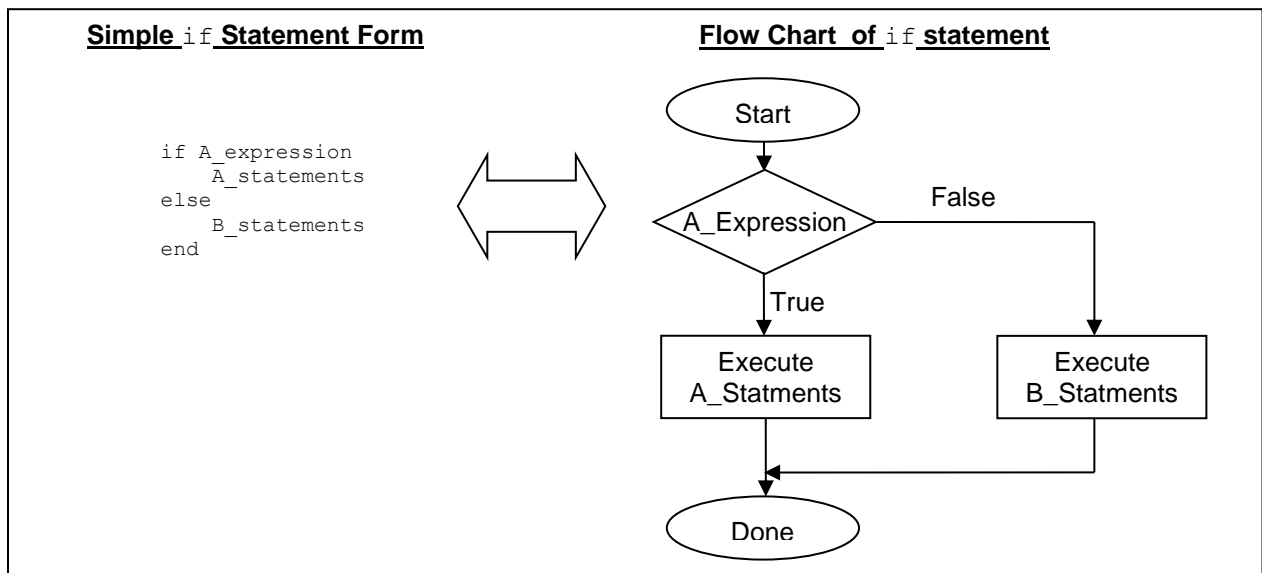
The program is executed sequentially by the computer unless a data flow control is used to redirect the program execution to a different location. Typically, data flow controls allow for a logical test using the relational operators to determine which statement should be executed.

The two most common data flow controls are the `if` conditional statement for decision making and the `for` loop statement to create a loop.

### if-else Conditional Statement

The statements following `if` will be executed if the conditional expression is true or non-zero. The `else` and `elseif` parts are optional; if you do not need them, then eliminate them. As long as there is an opening “`if`”, there has to be a terminating “`end`”.

The following figure shows the syntax of a simple “`if`” conditional statement and the execution flow using a flow chart.



The `A_Expression` is usually of the form

$(\text{exp1} <\text{operator}> \text{exp2})$   
where operators are relational operators such as `==`, `<`, `>`, `<=`, `>=`, `~=`.

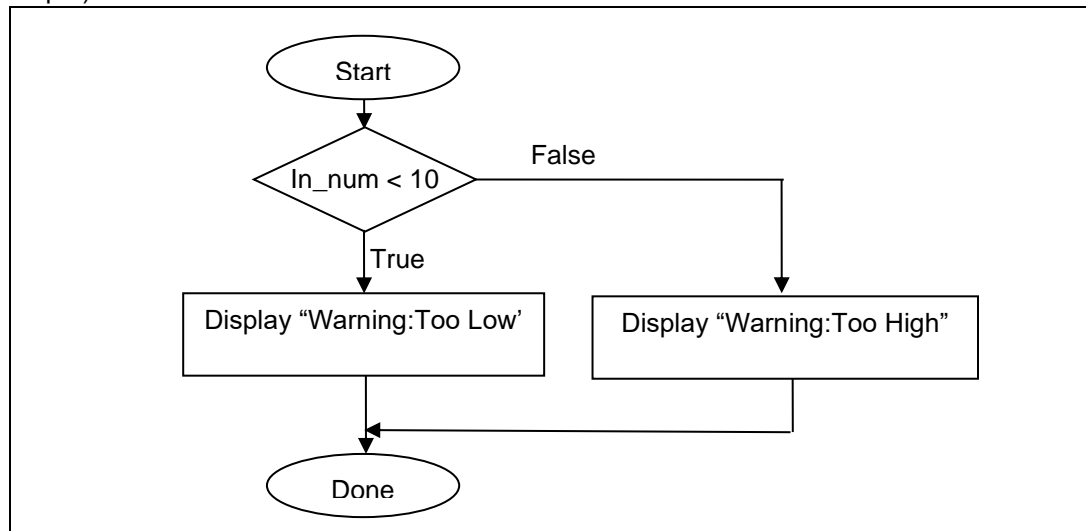
For example, `(a < b)` is a valid expression. Note that `A_Expression` can involve more than one relational operator.

### Example – if-else statement

Write a program that accepts one input. When the input is less than 10, it displays “Warning: Too Low”. Otherwise, it displays “Warning: Too High”.

### Solution

### Step 1) Translate Problem Statement to Flow Chart



### Step 2) Translate the flow chart to code

```
% Example- Use of if statement

function [ ] = if_example(in_num)
    if (in_num < 10)
        disp ('Warning: Too Low');    % Displays the string
    else
        disp ('Warning: Too High');
    end
    % end if (in_num <10)
end
```

### Student Exercise – if-else

Write a program that accepts two inputs; if the first input is larger than the second, the program displays the word “true”, and otherwise it displays “false”.

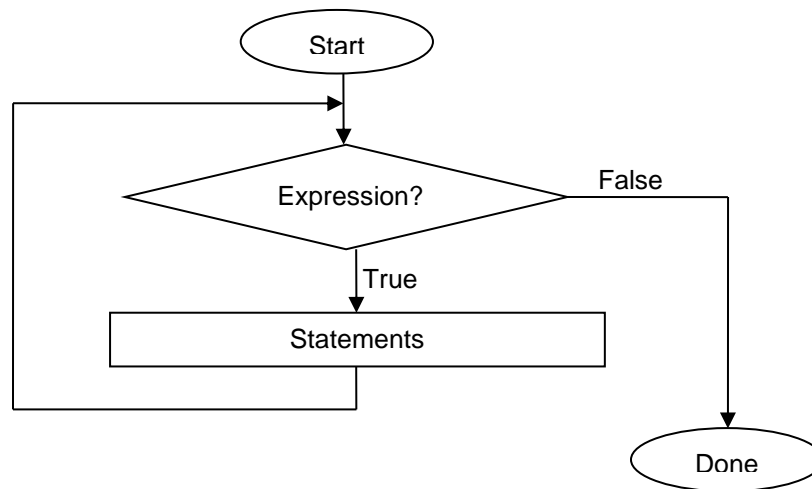
### Solutions

## while Loop Statement

The while loop construct repeats statements until the condition represented by the expression is not True.

```
while expression
  statements
end
```

The statements are executed while the expression evaluates to True. As shown by the following flow chart which is a graphical representation of the while loop flow:

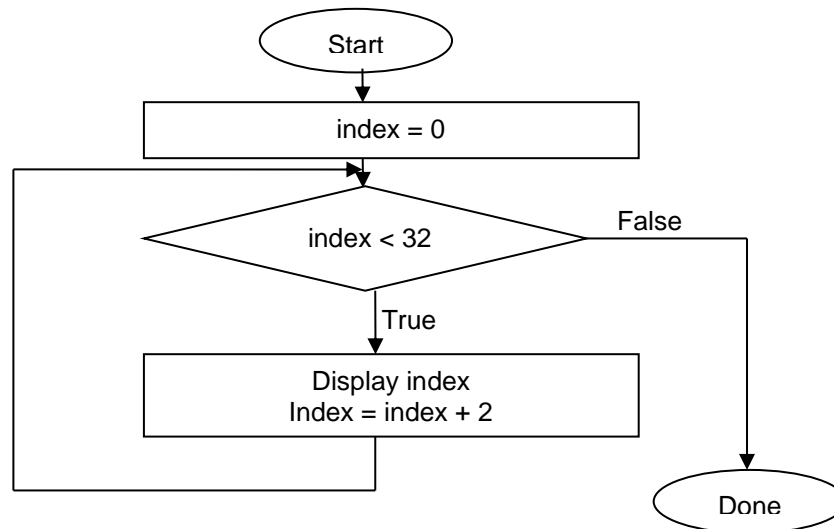


### Example – while loop

Display all the even numbers from 0 to 31.

### Solutions

Step 1) Translate the problem statement to flow chart.





Step 2) Translate the flow chart to code.

```
% Example- Use of while Loop

function [ ] = while_example ()
    index = 0;
    while (index < 31)
        % displays index since we do not have a semicolon
        % at the end of the line
        disp (index)
        index = index + 2;
    end % while Loop end statement
end
```

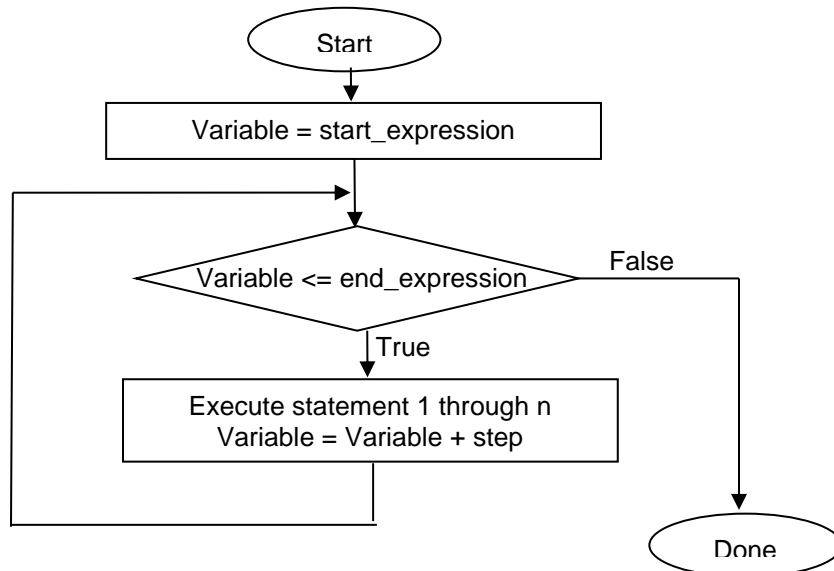
### for Loop Statement

The `for` loop statement repeats a set of statements a pre-determined number of times. The general form of the `for` loop statement is shown below:

```
for variable = start_expression: step : end_expression,
    statement1,
    ...
    statement n
end
```

This construct will execute statements 1 through `n` until `variable` is out of the range of `start_expression` to `end_expression`.

The following flow chart graphically represents the function performed by the `for` loop:



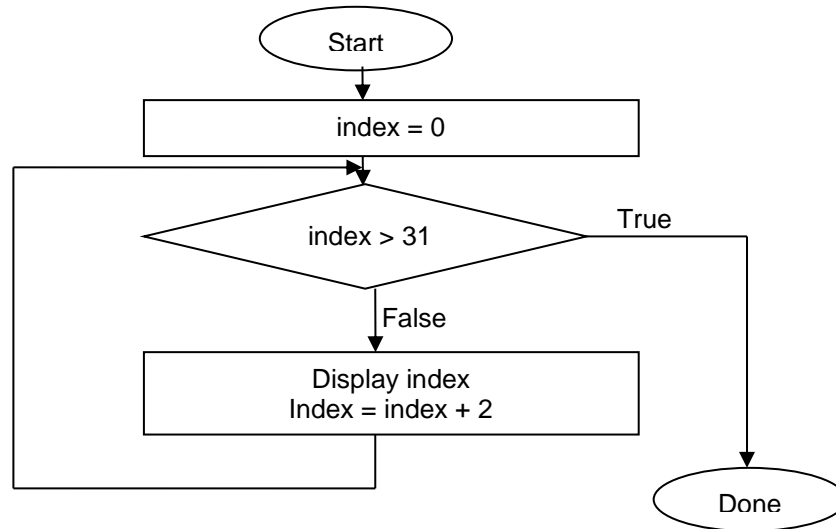
For example, the statement `for C=1:2:5` results in `C` taking on a value of 1 first time through the loop, a value of 3 the second time through the loop and a value of 5 the third time through the loop. It will *not*

go through the loop a fourth time because  $c$  would take on the value of 7, which is larger than 5, thus terminating the loop.

**Example – for Loop**

Display all the even numbers from 0 to 31.

Step 1) Translate the problem statement to flow chart



Step 2) Translate the flow chart to code

```
% Example- Use of for Loop  
  
function [ ] = for_example ()  
    for index = 0:2:31,  
        % displays index since we do not  
        % have ";" at then of line  
        index  
    end % for Loop end statement  
end
```

**Student Exercise – for Loop**

Write a program that displays all odd numbers from 5 to 35.

**Solution**

## Other Useful Functions

It is important to note that MATLAB has an extensive set of functionalities so use Help Section of MATLAB to search for new function. The following statement and functions are commonly used, so it may of use to the readers to review and understand their usage.

- **disp(value) – Display Raw Values**  
The function **disp(x)** will display the content of x.

For example, the following code segment displays “this is a test”:

```
X = 'This is a test' % note that ' is used to define string value.  
disp(X);
```

- **fprintf(fid, fmt, args...)** – **Display Formatted Text**  
The function **fprintf(fid, format\_string, A, ...)** formats the data according to the text in the variable `format_string` and display the information if `fid` is set to 1 or omitted.

The variable `format_string` can contain both ordinary characters and conversion specifications. Conversion specifications start with the character `%`, followed by a conversion character. Below is a list of valid conversion characters:

Specified	Description	Example
<code>%c</code>	Single character	D
<code>%d</code>	Signed decimal notation	-3.2
<code>%e</code>	Exponential notation (using a lowercase e as in 3.1415e+00)	-3.2e+00
<code>%f</code>	Fixed-point notation	3.567
<code>%o</code>	Unsigned Octal notation	234
<code>%s</code>	String	“testing”
<code>%u</code>	Unsigned decimal notation	10
<code>%x</code>	Hexadecimal notation	1FB

The special format strings `\n`, `\r`, `\t`, `\b`, and `\f` can be used to produce linefeed, carriage return, tab, backspace, and form feed characters, respectively. Use `\\` to produce a backslash character and `%%` to produce the percent character.

For example, the following code segment displays “This is 10 out of 23 pages”:

```
page_no = 10;  
total_page = 23;  
fprintf(1, 'This is %d out of %d pages. \n', page_no, total_page);
```

- **mod(x, y) – Modulo (Remainder) calculator**  
The function **mod(x, y)** calculates and returns the modulus, or integer(?) remainder, after x is divided by y.

For example, the statement `z = mod(125, 10)` sets the value of z to 5, since 5 is the remainder after dividing 125 by 10.

- **input(prompt) – Receive Typed User Input**  
The function **input(prompt)** displays the string prompt on the screen and waits for the user to type a response. Once the user presses Enter/Return, the value the user typed will be returned and program execution will continue.

For example, the code segment `x = input ('Enter a value:')` displays the string “Enter a value:” as a prompt for the user. Any number entered will be stored in the variable `x`.

- **tic and toc – Stopwatch Statements**

The `tic` and `toc` statements are commonly used to time the execution of a program. `tic` starts the stopwatch timer and saves the current time, and `toc` prints the time elapsed since then. To time a program, use `tic` and `toc` as shown below:

```
TIC
Operation % code to be timed
TOC
```

The above code displays the time used to complete the operation in seconds.

The functions presented here are a small introductory set with limited explanation. More information on presented functions and others are available on the *Help* section of MATLAB. To access the *Help* section, of MATLAB

### **Student Exercises – MATLAB**

Write a program that calculates the number of transistors if each transistor takes up 30 nm<sup>2</sup> on a device. The user will input the width and length of available silicon area of the device.

### **Solutions**

### **Student Exercises – MATLAB**

Write a program that allows the user to input current (in amps) and voltage (in volts). With each current and voltage entry, your program should calculate the remaining power from a 3 MW power source.

### **Solutions**

### **Student Exercises – MATLAB**

Draw a flow chart and write a program that calculates the next 20 Fibonacci numbers (following the sequence {1, 2, 3, 5, 8, 13, 21}).

### **Solutions**

## 8.7. Additional Resources

MathWorks. [MATLAB Reference Material Version R2000a](#). (2007) MathWorks

## 8.7. Problems

Instructions:

- MATLAB or free version (GNU Octave or SciLab) may be used for these problems.
- Homework solution should include flow chart and source code
- Be prepared to present your programs to the class

- 
1. List the steps from High level language to executable computer code.

---

  2. Write a program that displays the first 20 even numbers beginning with 100.

---

  3. Write a program that accepts three parameters “lower\_limit”, “upper\_limit” and “factor”. The program should display all the numbers between “lower\_limit” and “upper\_limit” that are evenly divisible by “factor”.

---

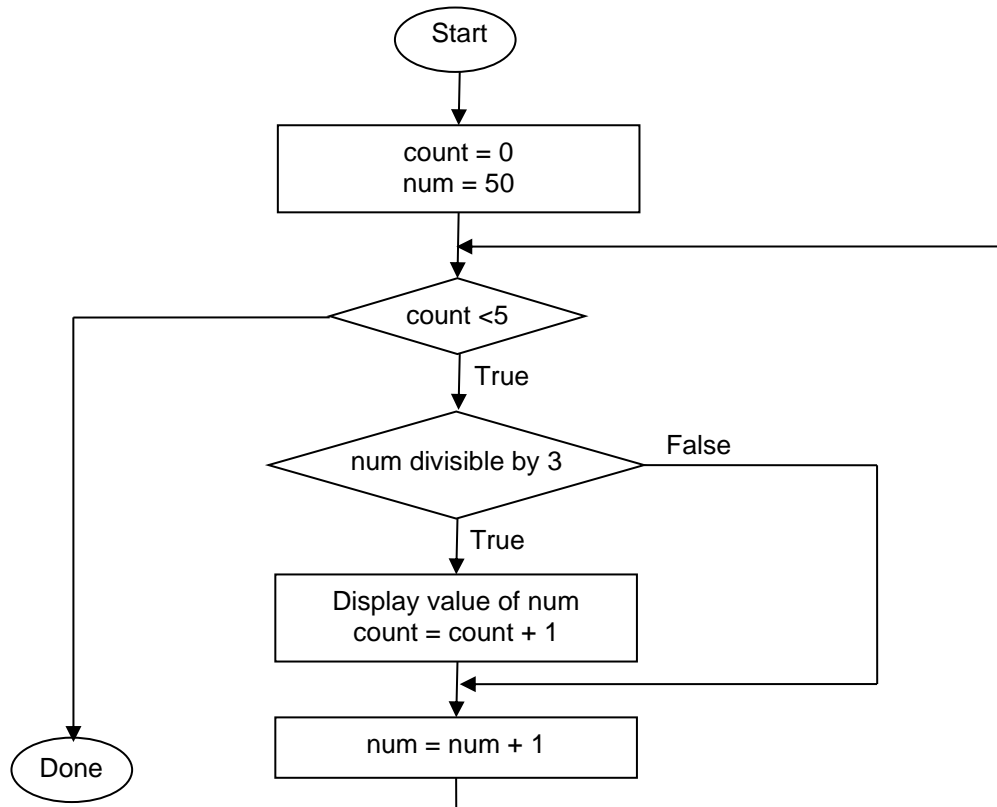
  4. Write a program that beeps every time the input parameter is evenly divisible by 13.  
*Hint: beep() function produces sound as long the speaker is not on mute.*

---

  5. Write a program that displays the 50 consecutive prime numbers starting with 2. Use MATLAB to display the CPU time required to execute this program.  
*Hint: tic() and toc() functions may be used. Standard Matlab functions such as prime() may not be used.*

---

  6. Write a program that implements the following flow chart:



---

7. Write a program that accepts a real number and displays the number in scientific format "x.xxxEn". For example 2546 would be displayed as 2.546000e+003

---

8. Write a program that accepts a fraction as two parameters "in\_numerator" and "in\_denominator". The program should remove all common factors between the two parameters and display "out\_numerator / out\_denominator" such that the value of fraction is maintained but there are no common factors between the "out\_numerator" and "out\_denominator".

---

9. Draw a flow chart and write the code for counting how many numbers from 10 to 149 is evenly divisible by 3.

---

10. Write a program that displays all numbers between input parameters "lower\_limit" and "upper\_limit" that have at least one digit equal to '2' .

*Note: lower\_limit and upper\_limit are positive integers.*



## **Chapter 9. Mathematical Concepts**

This Chapter cover a number of key mathematical concepts with significant applications to engineering. The concepts covered include introductory treatment of trigonometry, complex numbers, and matrices. Further, the focus of this Chapter is on the application of these mathematical concepts to ECS fields.

### **9.1. Key Concepts and Overview**

- Matrices
- MATLAB Matrix Operations
- Trigonometry
- MATLAB Trigonometry Operations
- Complex Numbers
- MATLAB Complex Number Operations

## 9.2. Matrices

Matrices are an important concept when a system has multiple input or output parameters, which is common in most systems. The terms *matrix* and *array* are often used interchangeably. **Matrices enable us to combine multiple inputs or output parameters into a single variable.** Use of matrices reduces the complexity of solving problems with multiple equations and unknowns, which are common in engineering.

### Matrix Format

A matrix is written with **its elements in rows and columns**. Each matrix element is referred to by its corresponding row and column. Below is the general form of a matrix:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}$$

$n$  columns  $\longrightarrow$   $\downarrow$   $m$  rows

- The above matrix is said to be an  $m \times n$  matrix, where  $m$  refers to the number of rows and  $n$  refers to the number of columns.
- In the general expression  $a_{k,j}$ ,  $k$  refers to the row and  $j$  refers to the column corresponding to the location of the element in the matrix  $a$ .

Vectors are one dimensional arrays:

- $A$  is called a **row vector** when  $m$  is equal to 1.

$$A = \left[ a_{1,1} \quad a_{1,2} \quad \dots \quad a_{1,n} \right]$$

- $B$  is called a **column vector** when  $n$  is equal to 1.

$$B = \begin{pmatrix} a_{1,1} \\ a_{2,1} \\ \vdots \\ a_{m,1} \end{pmatrix}$$

### Matrix Arithmetic

- **Adding Matrices**

Matrices can only be added if the matrices have exactly the same dimensions. The resulting matrix's elements are produced by the addition of the corresponding elements from the operand matrices. Elements of Matrix  $C$  which are the result of the addition of matrices  $A$  and  $B$  can be calculated as shown below:

$$C_{k,j} = A_{k,j} + B_{k,j}$$

Below is an example of adding two 2x3 matrices:

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 7 \end{pmatrix} + \begin{pmatrix} 6 & 10 & 8 \\ 9 & 12 & 11 \end{pmatrix} = \begin{pmatrix} 6+1 & 10+3 & 8+5 \\ 9+2 & 12+4 & 11+7 \end{pmatrix} = \begin{pmatrix} 7 & 13 & 13 \\ 11 & 16 & 18 \end{pmatrix}$$

- **Subtracting Matrices**

Subtracting has the **same requirements and process as addition**. Elements of Matrix  $C$  which are the result of subtraction of matrices  $A$  and  $B$  can be calculated as shown below:

$$C_{k,j} = A_{k,j} - B_{k,j}$$

Below is an example of subtracting two 2x3 matrices.

$$\begin{pmatrix} 10 & 13 & 5 \\ 2 & 14 & 7 \end{pmatrix} - \begin{pmatrix} 6 & 23 & 8 \\ 9 & 4 & 11 \end{pmatrix} = \begin{pmatrix} 10-6 & 13-23 & 5-8 \\ 2-9 & 14-4 & 7-11 \end{pmatrix} = \begin{pmatrix} 4 & -10 & -3 \\ -7 & 10 & -4 \end{pmatrix}$$

- **Scalar Multiplication**

A scalar is a single number value. In scalar multiplication, the scalar value is multiplied by all the elements in the matrix. So the scalar value  $a$  is multiplied by matrix  $B$ , resulting in a matrix  $C$  where each element is calculated by:

$$C_{k,j} = a \times B_{k,j}$$

Below is an example of 2x2 matrix multiplied by a scalar of 4:

$$4 * \begin{pmatrix} 6 & 23 \\ 9 & 4 \end{pmatrix} = \begin{pmatrix} 4*6 & 4*23 \\ 4*9 & 4*4 \end{pmatrix} = \begin{pmatrix} 24 & 92 \\ 36 & 16 \end{pmatrix}$$

- **Matrix Multiplication**

Multiplication of two matrices **requires that the number of columns of the first matrix be equal to the number of rows in the second matrix**. Matrix multiplication is not commutative ( $A \times B$  does not necessarily equal  $B \times A$ ) and not all matrices can be multiplied.

Elements of  $C$ , the resulting matrix from multiplication of two matrices  $A$  and  $B$ , where  $A$  is a  $p \times n$  matrix and  $B$  is a  $n \times q$  matrix, can be calculated using:

$$c_{i,j} = \sum_{k=1}^n a_{i,k} * b_{k,j} \quad \text{More common English explanation?}$$

It is also important to note that the resulting matrix will have dimensions of  $p \times q$ .

- **Example** - Multiplication of 3x2 matrix by 2x3 matrix

Note: The number of columns of first matrix and the number of rows of the second matrix are the same (2); therefore, the resulting matrix will be a square matrix 3x3.

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} * \begin{pmatrix} 10 & 30 & 50 \\ 20 & 40 & 60 \end{pmatrix} = \begin{pmatrix} (1*10 + 4*20) & (1*30 + 4*40) & (1*50 + 4*60) \\ (2*10 + 5*20) & (2*30 + 5*40) & (2*50 + 5*60) \\ (3*10 + 6*20) & (3*30 + 6*40) & (3*50 + 6*60) \end{pmatrix}$$
$$= \begin{pmatrix} 90 & 190 & 290 \\ 120 & 260 & 400 \\ 150 & 330 & 510 \end{pmatrix}$$

### 9.3. MATLAB Matrix Operations

MATLAB offers a complete set of matrix operations. In addition to the operations already discussed, such as addition, multiplication, and division, the following functions are worthy of consideration:

#### Creating a Matrix

A matrix can be entered into MATLAB by entering each row element separated by a comma (,) and each row is separated by semicolon (;), all surrounded by square brackets ([ ]).

For example, to enter  $A = \begin{pmatrix} 6 & 8 \\ 9 & 4 \end{pmatrix}$ , type in the following command:

```
>> A = [6, 8; 9, 4]
```

#### Matrix with rows and columns of zeros

The command `zeros(r,c)` creates a matrix that contains  $r$  rows and  $c$  columns of 0s.

To create a 3x4 matrix of zeros:  $A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

type in the following command:

```
>> A = zeros(3, 4)
```

#### Matrix with rows and columns of ones

The command `ones(r,c)` creates a matrix that contains  $r$  rows and  $c$  columns of 1s.

To create a 3x2 matrix of ones:  $A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$

Type in the following command:

```
>> A = ones(3, 2)
```

#### Custom Matrix Definition

MATLAB allows the definition of a matrix by entering values or using the “zeros” and “ones” commands.

For example, to create the matrix  $A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 \\ 9 & 9 & 9 & 9 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Type in the following command

```
>> A = [2, 3, 4, 5; ones(1,4); 9 * ones(1,4); zeros(1,4)]
```

### Matrix creation by indexing

MATLAB offers the ability to create a matrix by indexing. The following MATLAB command is used to create a matrix by indexing.

```
% starts with N and increments by K each time  
% Continues as long as the value is less than or equal to M.  
A = [N : K : M]
```

For example, the command

```
>> A = [3 : 2 : 9]
```

Returns the matrix

```
[3, 5, 7, 9]
```

### Accessing Matrix Elements

Matrix elements are identified by their location in the matrix in terms of their **row and column number**. It is important to note that row and column numbers **start with 1** and are **always positive integers**.

Each element of a matrix can be accessed by using the name of the array and the row and column numbers. For example, in the following array:

$$A = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{pmatrix}$$

the element in row 3 and column 2 is accessed by the following command:

```
>> A(3,2)
```

MATLAB also allows access to a range of elements in a matrix. For example, to access the elements from row 2 to 3 and column 1 to 4, use the following command:

```
>> A(2:3, 1:4)
```

The above command returns a 2x4 matrix shown below:

$$\begin{pmatrix} 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{pmatrix}$$

### Term-by-Term Operators

Term-by-term operators perform the same operation on each element and save the result in the corresponding elements of the resultant matrix. They are `.*` for multiplication, `./` for division, and `.^` for power.

Given matrices

$$A = \begin{pmatrix} 6 & 8 \\ 9 & 4 \end{pmatrix}, B = \begin{pmatrix} 1 & 3 \\ 2 & 5 \end{pmatrix}$$

Using the following MATLAB command

```
>> C = A .* B
```

Evaluates  $C = A * B = \begin{pmatrix} 6*1 & 8*3 \\ 9*2 & 4*5 \end{pmatrix} = \begin{pmatrix} 6 & 24 \\ 18 & 20 \end{pmatrix}$

### Systems of Equations

Matrices are useful for solving systems of equations. By writing systems of equations in matrix form, we can take advantage of MATLAB functions to solve them.

For example, a system of equations can be transformed as shown below:

$$\begin{cases} 2x + 3y + 8z + 7w = 1 \\ 3x - 8y + 2z + 9w = -3 \\ 5x + 4y - 4z + 5w = 5 \\ x + 3y - 6z + 8w = 10 \end{cases} \Rightarrow \begin{pmatrix} 2 & 3 & 8 & 7 \\ 3 & -8 & 2 & 9 \\ 5 & 4 & -4 & 5 \\ 1 & 3 & -6 & 8 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 1 \\ -3 \\ 5 \\ 10 \end{pmatrix}$$

Now, if we make the following assignments:

$$A = \begin{pmatrix} 2 & 3 & 8 & 7 \\ 3 & -8 & 2 & 9 \\ 5 & 4 & -4 & 5 \\ 1 & 3 & -6 & 8 \end{pmatrix}, \quad X = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}, \quad C = \begin{pmatrix} 1 \\ -3 \\ 5 \\ 10 \end{pmatrix}$$

The original system of equations can be rewritten as:

$$A \times X = C$$

Multiply both sides with the inverse of  $A \rightarrow A^{-1}$

$$X = A^{-1} \times C$$

By writing a program in MATLAB that describes the above equation, we will be able to solve for the unknown  $X$ . Here are the commands

```
C = [1; -3; 5; 10];
A = [2, 3, 8, 7; 3, -8, 2, 9; 5, 4, -4, 5; 1, 3, -6, 8];
X = A^(-1) * C
```

which return the value of the unknown  $X$ :

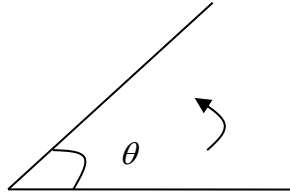
$$X = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} -0.67 \\ 0.72 \\ -0.55 \\ 0.65 \end{pmatrix} \rightarrow x=-0.67, y=0.72, z=-0.55, w=0.65$$

## 9.4. Trigonometry

Trigonometry is a branch of mathematics that deals with relationships between distances and angles. This section discusses the phase or angle, followed by the four main trigonometric functions: *sine*, *cosine*, *tangent*, and *cotangent*.

### Angle or Phase

An angle or phase measures the rotation of a line compared to another as shown by  $\theta$  in the following diagram:



### Measurements

Degrees and radians are the two most commonly used units for measuring angles.

Degrees are distinguished with the degree symbol ( $^\circ$ ) after the value. A circle goes from  $0^\circ$  to  $360^\circ$  for a full counter-clockwise rotation before repeating.

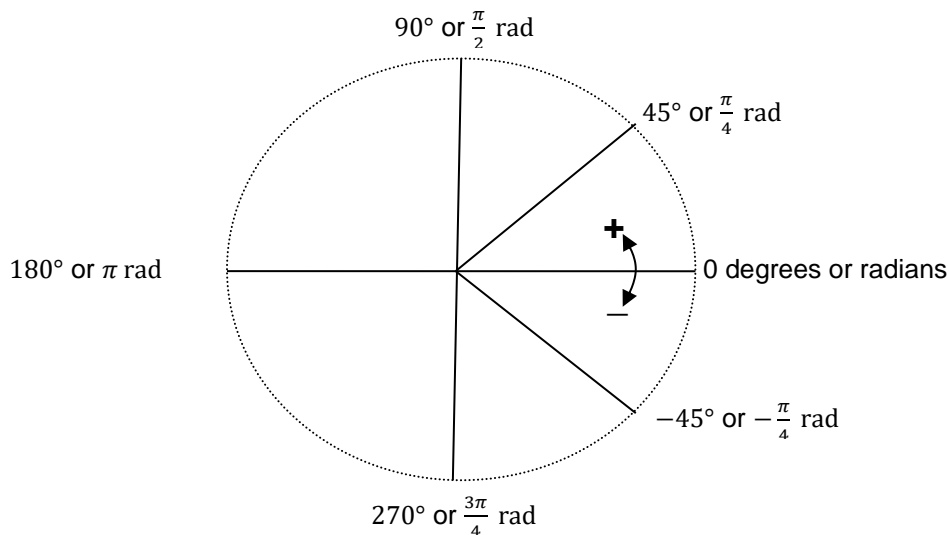
Radian measures go from 0 to  $2\pi$  radians for a full counter-clockwise rotation. There is no symbol to distinguish radians, although rad is sometimes used.

Degrees and radians may be converted using the following relationships:

$$\text{Radians} = \frac{2\pi}{360^\circ} \times \frac{\text{Degrees}}{1}$$

$$\text{Degrees} = \frac{360^\circ}{2\pi} \times \frac{\text{Radians}}{1}$$

Refer to the following diagram for examples of various angles:



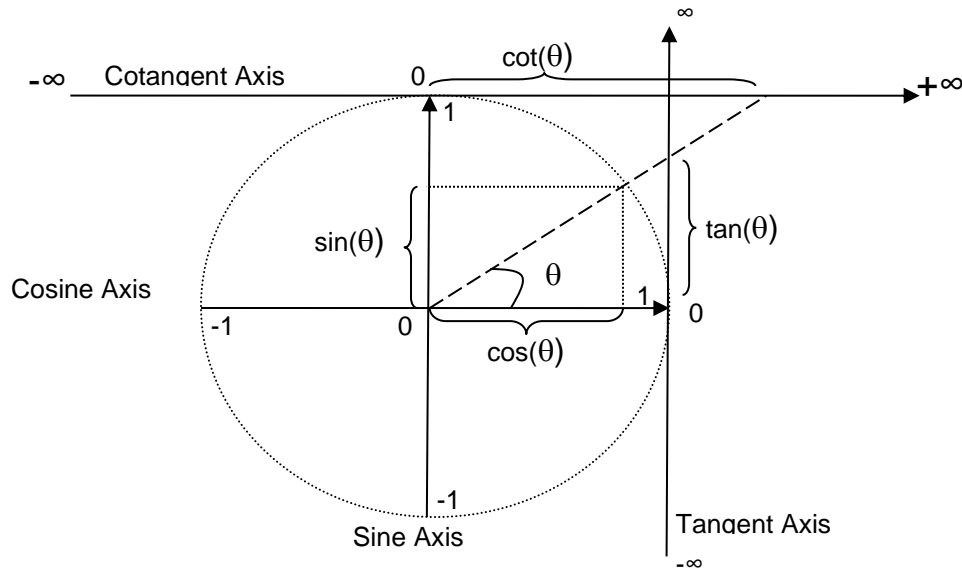


As shown in the diagram, counter-clockwise rotation is positive and clockwise rotation is negative.

### Trigonometry Functions

There are four trigonometry functions commonly used in engineering and science fields: sine, cosine, tangent, and cotangent. Each of these functions shows a relationship between the angles and sides of a right triangle.

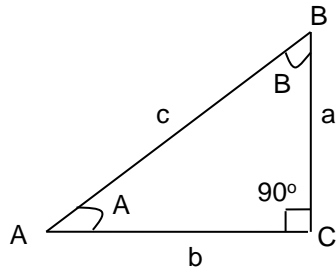
Typically, a unit circle (a circle with radius equal to 1) is used to show the results of each of the four trigonometry functions.



Each function has specific upper and lower limits as shown in the following table:

	Maximum	Minimum
$\sin \theta$	1	-1
$\cos \theta$	1	-1
$\tan \theta$	$\infty$	$-\infty$
$\cot \theta$	$\infty$	$-\infty$

The next step is to define each of the four trigonometric functions in terms of the associated sides of a right triangle.



$$\sin(A) = \frac{\textit{Opposite}}{\textit{Hypotenuse}} = \frac{a}{c}$$

$$\cos(A) = \frac{\textit{Adjacent}}{\textit{Hypotenuse}} = \frac{b}{c}$$

$$\tan(A) = \frac{\textit{Opposite}}{\textit{Adjacent}} = \frac{a}{b}$$

$$\cot(A) = \frac{\textit{Adjacent}}{\textit{Opposite}} = \frac{b}{a}$$

### Sine and Cosine Relationships

- Pythagorean Identity

$$\textit{Pythagorean Theorem} \Rightarrow a^2 + b^2 = c^2$$

$$\textit{Divide by } c^2 \Rightarrow a^2 / c^2 + b^2 / c^2 = c^2 / c^2$$

$$\textit{Substitute sin \& cos} \Rightarrow \sin^2 \theta + \cos^2 \theta = 1 \quad \text{"Pythagorean Identity"}$$

- Side angles

$$\sin(A) = \cos(B) = \frac{a}{c}$$

$$\cos(A) = \sin(90^\circ - A)$$

- Tangent and Cotangent Relationships

$$\tan(A) = \frac{\sin(A)}{\cos(A)}$$

$$\cot(A) = \frac{\cos(A)}{\sin(A)}$$

$$\cot(A) = \frac{1}{\tan(A)}$$

- Other Useful Relationships**

$$\sin(A \pm B) = \sin(A)\cos(B) \pm \sin(B)\cos(A)$$

$$\cos(A \pm B) = \cos(A)\cos(B) \mp \sin(A)\sin(B)$$

- Power Series**

**Computers use the power series approach to calculate trigonometry functions.** These series were developed by Newton and other mathematicians in the 17<sup>th</sup> century.

Here are the power series for angle x (where x is in radians):

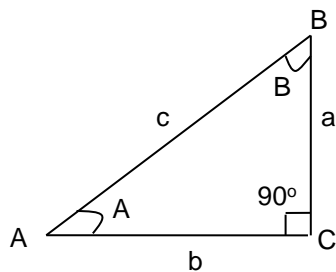
$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{(2n+1)}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{2n!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots$$

- **Inverse Functions**

Any one of the trig functions can be derived from any other trig function.

Additionally, **each function has an inverse function** where if the value of trig function was known then the inverse function provides us with the angle. Inverse function are denoted by either a superscript “-1” or arc in front of the function.



$$\arcsin\left(\frac{a}{c}\right) = \sin^{-1}\left(\frac{a}{c}\right) = A$$

$$\arccos\left(\frac{b}{c}\right) = \cos^{-1}\left(\frac{b}{c}\right) = A$$

$$\arctan\left(\frac{a}{b}\right) = \tan^{-1}\left(\frac{a}{b}\right) = A$$

$$\text{arccot}\left(\frac{b}{a}\right) = \cot^{-1}\left(\frac{b}{a}\right) = A$$

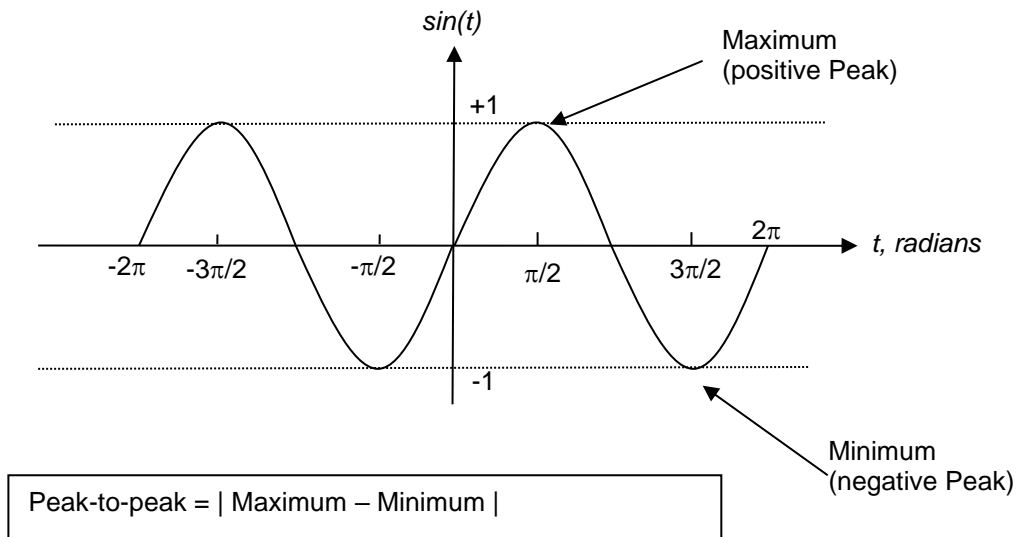
### Application of Trigonometry in Describing Waveforms

When the value of an angle changes, the value of the angle's sine changes correspondingly. Electrical signals, sound waves and even light waves travel in time corresponding with the shape of a sine wave (sinusoidal signal).

Below are the graphs of a sine wave when angle is equal to  $t$ ,  $wt$ , and  $wt + \phi$ .

#### Sine Graph when Angle is $t$

In this case, let  $t$  be in radians. We can plot  $\sin t$  as shown below:



### Amplitude - Sine Graph when Angle is $wt$

In this case, we are redefining the angle with parameters that are commonly used to describe electrical, audio and light waves.

The following is a plot of  $\sin wt$  where:

$t$  represents time in seconds

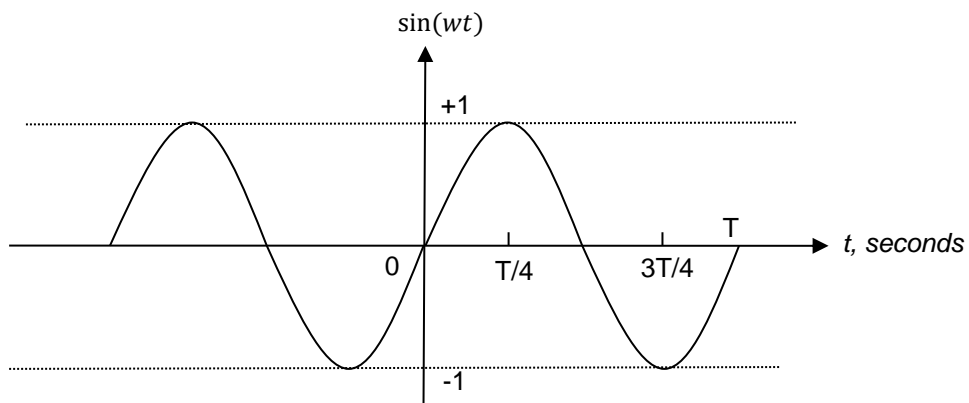
$w$  represents angular frequency in radians/second

$w = 2f\pi = \frac{2\pi}{T}$  where:

$f$  represents the frequency in Hertz or cycles/second

$T$  represents the period in seconds

A period is the time required to complete one of the repeating patterns.



### AC Electrical Signal coming to homes in the United States

When viewing the signal voltage on the oscilloscope, the signal observed is similar to the one shown above. The frequency of AC signal that comes to homes is 60 hertz ( $f = 60$ ).

We can use the  $f = \frac{1}{T}$  relationship to calculate the signal period of  $T = \frac{1}{60}$  seconds.

### Phase Shifting - Sine Graph when Angle is $wt \pm \phi$

In order to shift the sine wave over time, we simply can add a positive or negative constant (phase shift) to move it to the left or right correspondingly. The constant  $\phi$  represents the phase shift.

The following is the plot of  $\sin(wt \pm \phi)$  where:

$t$  represents time in seconds

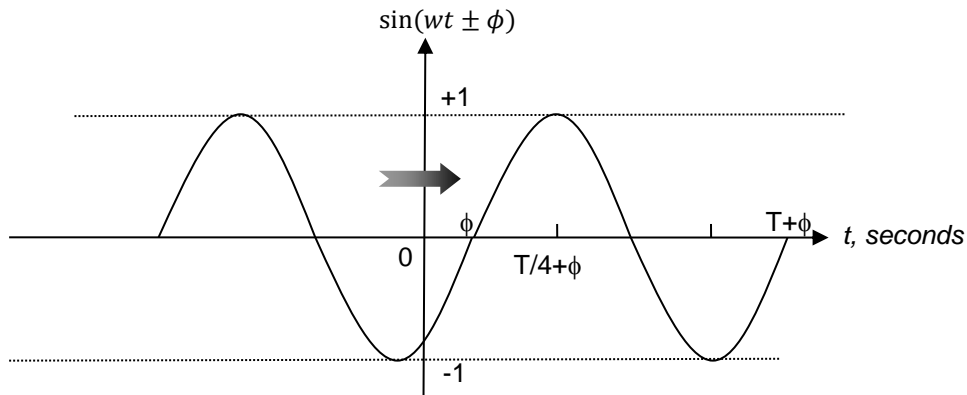
$w$  represents angular frequency in radians/second

$w = 2\pi f = \frac{2\pi}{T}$  where:

$f$  represents the frequency in Hertz or cycles/second

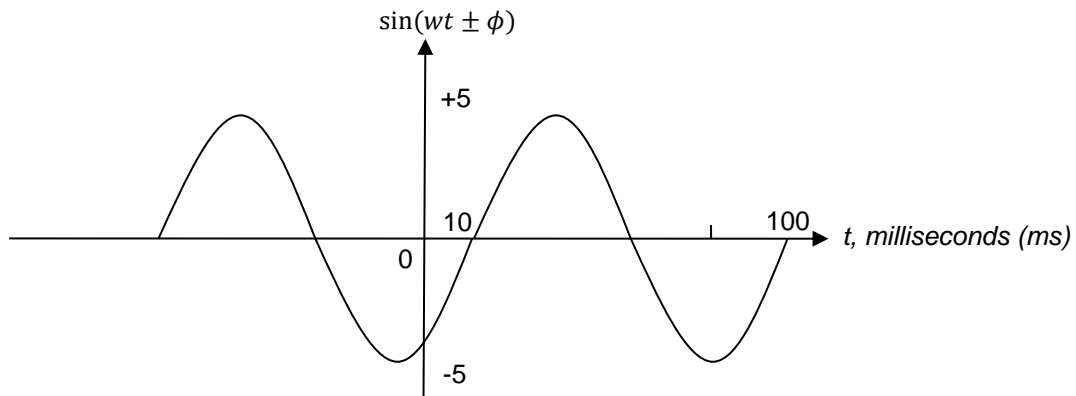
$T$  represents the period in seconds

$\phi$  represents the phase shift in radians.



### Student Exercise – Trigonometry

Write the function  $f(t)$  that is represented by the following diagram



### Solution

## 9.5. MATLAB Trigonometry Operations

Sinusoidal signals play an important role in modeling and analyzing many engineering and scientific systems. MATLAB's trigonometry functions and its graphing capabilities are useful in analyzing/designing time- and frequency-dependent sinusoidal systems.

### Calculating Trigonometry Functions

By default, MATLAB assumes that angles are in radians ( $\pi$  radians = 180 degrees), so remember to convert all angles to radians. Another useful fact is that MATLAB uses the function `pi()` to return the value of  $\pi$ .

MATLAB can calculate all of the major trigonometric functions. Below are the most useful ones:

- **Sine,  $y = \sin(x)$ , and Inverse Sine,  $x = \text{asin}(y)$**   
Examples:
  - (a) `sin(pi/4)` returns 0.707
  - (b) `asin(0.707)` returns  $0.7852 = (3.14)/4 = \pi/4$
- **Cosine,  $y = \cos(x)$ , and Inverse Cosine,  $x = \text{acos}(y)$**   
Examples:
  - (a) `cos(pi/3)` returns 0.5
  - (b) `acos(0.5)` returns  $1.0467 = (3.14)/3 = \pi/3$
- **Tangent,  $y = \tan(x)$ , and Inverse Tangent,  $x = \text{atan}(y)$**   
Examples:
  - (a) `tan(pi/4)` returns 1.0
  - (b) `atan(1.0)` returns  $0.7852 = (3.14)/4 = \pi/4$

### Application of Matrices

MATLAB functions **operate on an array or matrix the same way as they operate on scalar values**. These features are useful if we have a set of data to analyze.

#### Example – Matrices

Calculate the value of sine from  $-\pi$  to  $+\pi$  in  $\pi/10$  increments.

```
% angle is an array containing  $-\pi, -9\pi/10, \dots, 9\pi/10, \pi$ 
angle=[-pi : pi/10 : +pi];

% sine-value is an array where each element is the sine
% of corresponding element in angle array.
sine_value = sin(angle);
```

Executing the above commands results in `sine_value` containing 21 elements which are shown below:

```
sine_value = [-0.0000, -0.3090, -0.5878, -0.8090, -0.9511, -1.0000,
-0.9511, -0.8090, -0.5878, -0.3090, 0, 0.3090, 0.5878, 0.8090, 0.9511,
1.0000, 0.9511, 0.8090, 0.5878, 0.3090, 0.0000]
```

### Graphing

MATLAB offers extensive graphing capabilities, including two- and three- dimensional plots as well as animated graphs. We will be focusing on the 2-dimensional capabilities, but students are encouraged to explore additional MATLAB capabilities shown in the graphics section demos.

The two foundational graphics functions are plot and stem.

### Plot Points – `plot(x, y)`

The plot function enables the designer to plot a two-axis line plot.

```
% plots vector y on vertical axis versus vector x in horizontal axis.  
% (a one dimensional matrix (1xn or nx1) is referred to as a vector)
```

```
plot (x, y)
```

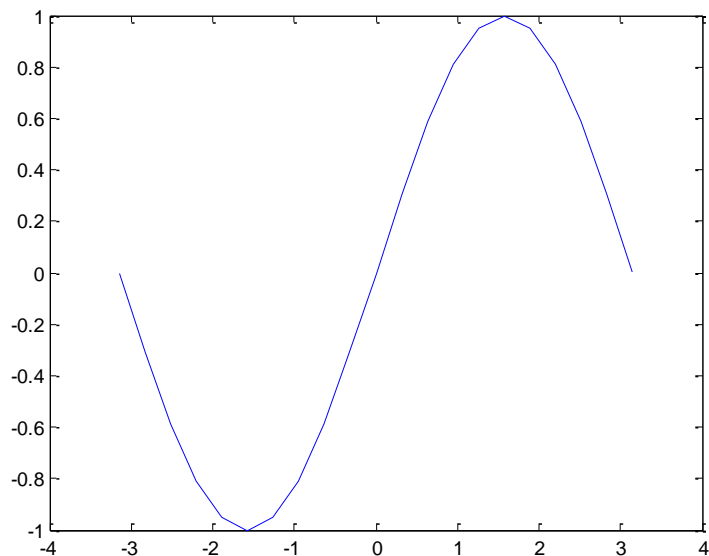
### Example – plotting

Calculate and plot the value of sine from  $-\pi$  to  $+\pi$  in  $\pi/10$  increments.

### Solution

```
% angle is an array containing  $-\pi, -9\pi/10, \dots, 9\pi/10, \pi$   
angle=[-pi : pi/10 : +pi];  
  
% sine-value is an array where each element is the sine  
% of the corresponding element in the angle array.  
sine_value = sin(angle);  
  
% function to plot sine_value versus angle.  
plot (angle, sine_value);
```

The resulting plot is shown below:



## Annotating the Graph

When graphing data, it is important that the axes be clearly labeled and the graph title be specified. Additionally, it may be important to use a specific color and specific markers for each set of data. MATLAB allows the user to make all of these selections and more. For complete instructions on these and other graphing techniques, refer to the *Help* section of MATLAB.

As a means to demonstrate some of these features, let's do another example that adds more information to the above graph:

### Example – Annotation

Calculate and plot the value of sine from  $-\pi$  to  $\pi$  in  $\pi/10$  increments. Label the axes and title the graph.

### Solution

The following code is a full function that performs the calculation and plotting.

```
% full_plot.m
% Author: Izad Khormaei
% Last update: 9/10/06, Version: 1.1
% Input parameters: n/a
% output parameter: n/a
% Description: Calculates and plots the value of sine from  $-\pi$  to  $+\pi$  in  $\pi/10$  increments.
% It will also label the axes and title of the plot.

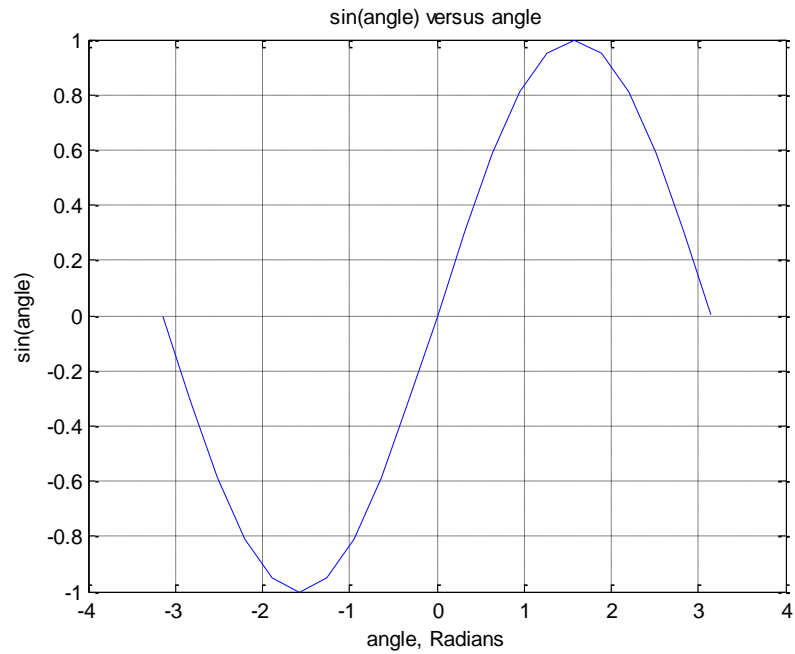
function [] = full_plot ()
    % Calculate the angle and sine
    angle=[-pi : pi/10 : +pi]; % angle is an array containing  $-\pi, -9\pi/10, \dots, 9\pi/10, \pi$ 
    sine_value = sin(angle); % sine-value is an array where each element is the sine
    % of the corresponding element in the angle array.

    plot (angle, sine_value); % plot sin_value versus the angle

    % Configure the plot
    title('sine(angle) versus angle') % Add title to the plot
    xlabel('angle, Radians') % Add horizontal axis label
    ylabel('sine(angle)') % Add vertical axis label
end % return full_plot()
```

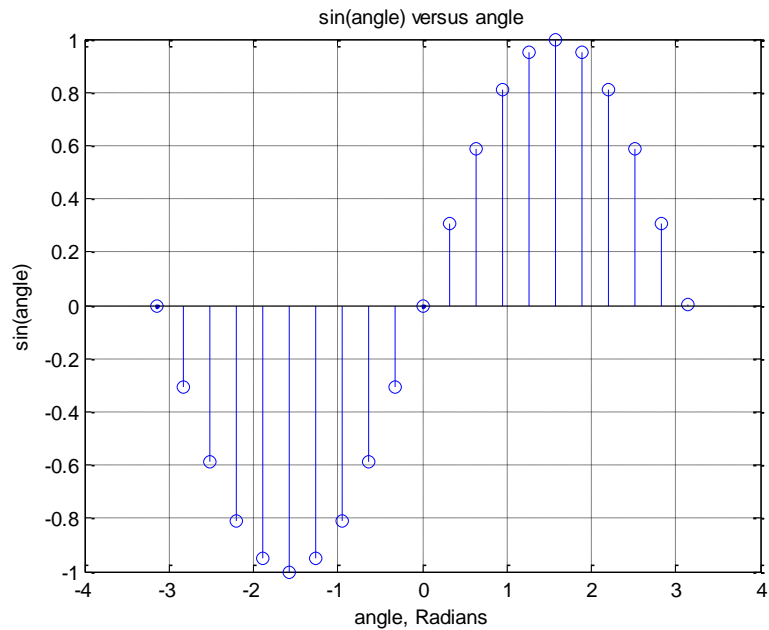


The resulting plot is shown below:



### Plot Stem Chart – stem(x, y)

If the function plot() is replaced with stem(), then we will see the results in a stem or discrete form, as shown below:



### Other Plotting Functions

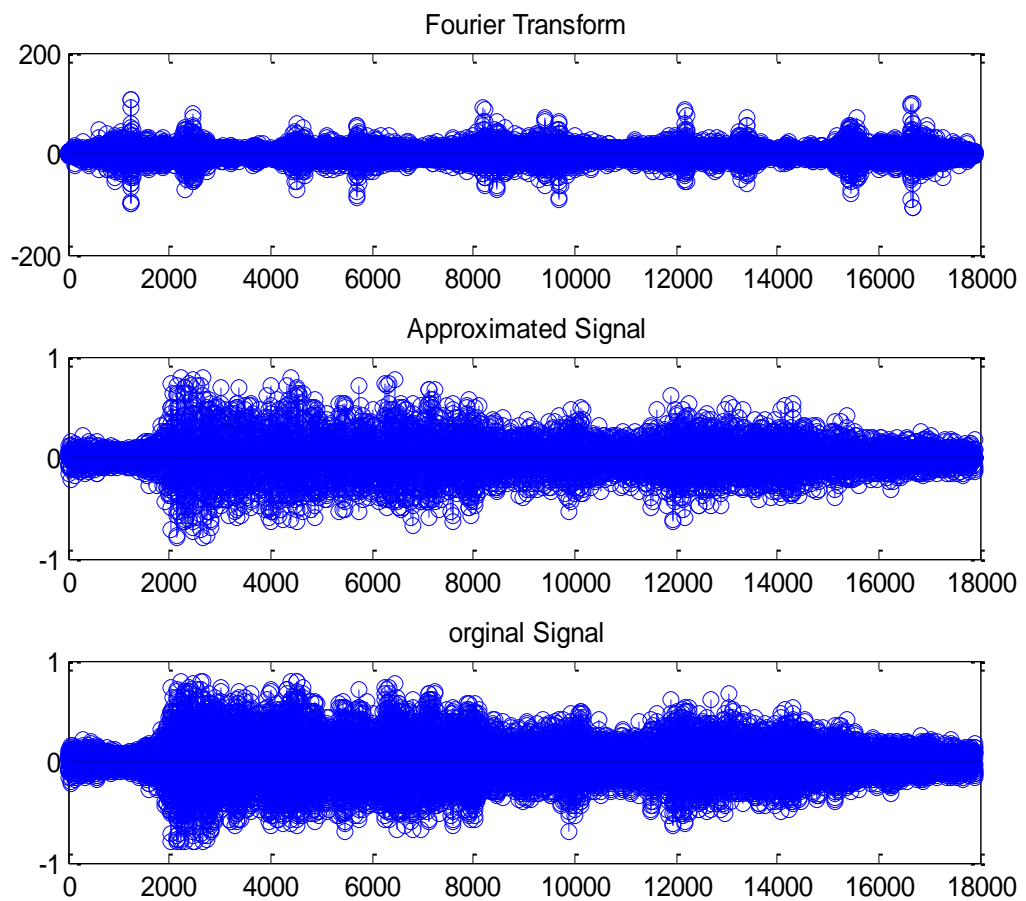
As mentioned earlier, there is an extensive level of graphics functionality available in MATLAB which would be valuable to explore in the future. The `subplot` function enables the user to plot multiple plots in a grid.

`subplot(m,n,p)` creates a grid of  $m$ -by- $n$  matrix and places the next plot or stem graph in the  $p^{\text{th}}$  cell of the matrix. The cells are counted along the top row of the Figure window, then the first column of the second row, and so on.

For example, the following code generates a figure that contains three rows by one column of graphs:

```
subplot(3,1,1), stem(X), title(' Fourier Transform') % first plot row 1, col 1
subplot(3,1,2), stem(x), title(' Approximated Signal') % second plot row 2, col 1
subplot(3,1,3), stem(y(100:18000)), title(' original Signal'); % third plot row 3, col 1
```

The result of this code is shown in the following figure.



## 9.6. Complex Numbers

Complex numbers came into existence in response to the fact that mathematicians needed to use  $\sqrt{-1}$  to solve quadratic equations. But in the real number paradigm,  $\sqrt{-1}$  is an invalid number, since there is no real number that when squared would equal  $-1$ .

In the 18<sup>th</sup> century, mathematicians came up with a solution to this dilemma by creating a new entity and referring to it as  $i$ . In engineering, we will use  $j$  to refer to this new entity since  $i$  is reserved for identifying current.

$j$  is conveniently defined as:

$$j = \sqrt{-1} \quad \text{and} \quad j^2 = -1$$

### Complex Number General Form

The general form of a complex number has a real part and an imaginary part and is represented as:

$$C = x + jy \quad \text{where:}$$

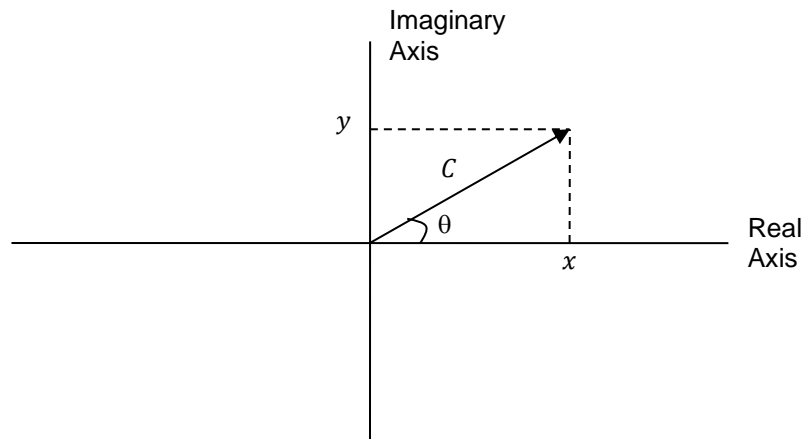
$x$  is a real number constant and is referred to as the real part of  $C$

$y$  is a real number constant and is referred to as the imaginary part of  $C$ .

- For example, the complex number  $2 - 6.2j$  has a real part equal to 2 and an imaginary part equal to  $6.2j$ .

### Plotting Complex Numbers

Complex numbers are typically plotted on a plane where the vertical axis represents imaginary numbers and the horizontal axis represents real numbers. The following diagram depicts a vector representing the complex number  $C = x + jy$ :



### Relationships

Here we can **use trigonometric functions to describe relationships** between various components such as the phase (angle,  $\theta$ ), real part, imaginary part and magnitude. For a complex number  $x + jy$ , the relationships are described below:

- The **length of the complex number vector is called its magnitude**. The complex number magnitude is defined by the following equation which uses the right triangle properties:

$$\text{magnitude of } C = |C| = \sqrt{x^2 + y^2}$$

- The phase or angle of a complex number is the angle  $\theta$  between the real axis and the complex number vector. Utilizing the trigonometric functions,  $\theta$  is defined by:

$$\theta = \arctan \frac{\text{Imaginary Part}}{\text{Real Part}} = \arctan \frac{y}{x} \quad \text{or} \quad \theta = \arcsin \frac{y}{|C|}$$

### Complex Number Arithmetic

- **Addition and Subtraction**

All the rules of real number arithmetic apply here, but it is mandatory to keep the imaginary and real parts separated. Add/subtract all the real numbers (real parts) and then add/subtract all of the coefficients of  $j$  (Imaginary parts). Below is an example:

$$(2 + j3.5) - (2.3 - j4.1) = (2 - 2.3) + j(3.5 + 4.1) = -0.3 + j7.6$$

- **Multiplication**

All of the rules of real number arithmetic apply here, and additionally, the definition  $j^2 = -1$  applies. Below is an example:

$$(2 + j4) \times (3 - j5) = (2 \times 3) - j(2 \times 5) + j(4 \times 3) - (4 \times 5)(j2) = 6 - j10 + j12 - (20)(-1) = 26 + j2$$

- The general process of multiplication can be shown as:

$$(x + jy)(u + jv) = (xu - yv) + j(xv + yu)$$

### Complex Number Applications

Leonard Euler (pronounced "Oiler"), an influential 18<sup>th</sup> century Swiss mathematician and physicist, put forth the following relationship which has come to be known as Euler's Identity:

$$e^{ja} = \cos a + j \sin a$$

This is a very important equation in engineering since it is the general representation of sinusoidal signals as shown below:

$$\text{real part}\{e^{ja}\} = \cos a \quad \text{since } \sin(a) \text{ is the imaginary part}$$

In electrical engineering, devices such as capacitors and inductors change their behavior with **signal** frequency. Use of  $e^{ja}$  allows us to design and analyze circuits with time-varying and frequency-sensitive elements without needing to use calculus and differential equations.

There are two other ways of writing Euler's Identity which may prove useful:

$$\cos a = \frac{e^{ja} + e^{-ja}}{2} \quad \& \quad \sin a = \frac{e^{ja} - e^{-ja}}{j2}$$

## 9.7. MATLAB Complex Number Operations

MATLAB operators support complex numbers, so all arithmetic operators can be used for complex numbers. MATLAB accepts both  $i$  and  $j$  as the representations of the imaginary ( $j=i=\sqrt{-1}$ ), but it defaults to  $i$  when displaying the results. For example:

$$C=(1 + j2)*(1 - j3) = 7 - j$$

MATLAB also provides the function “exp(x)” which evaluates  $e^x$ . Moreover, the following function proves useful when working with complex numbers:

- **imag (C)**  
This function returns the imaginary portion of a complex number “C”. For example:  

```
>> imag(10 + j12)
```

returns the value 12, which is the imaginary part.
- **real (C)**  
This function returns the real portion of a complex number “C”. For example:  

```
>> real(10 + j12)
```

returns the value 10, which is the real part.
- **abs (C)**  
This function returns the magnitude or absolute value of complex number “C”. For example:  

```
>> abs(10 + j12)
```

returns the value 15.62 which is  $|10 + j12| = \sqrt{10^2 + 12^2} = 15.62$
- **angle (C)**  
This function returns the phase or angle of complex number “C”. For example:  

```
>> angle (10 + j12)
```

return the value 0.88 which is the phase of the complex number  $= \tan^{-1} \frac{12}{10}$ .

## 9.8. Additional Resources

- MathWorks. [MATLAB Reference Material Version R2000a](#). (2007) MathWorks

## 9.9. Problems

Instructions:

- MATLAB or free version (GNU Octave or SciLab) may be used for these problems.
- Homework solution should include flow chart and source code
- Be prepared to present your programs to the class

---

1. For the following two matrices

$$A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 1 & 0 & 0 & 3 \\ 7 & 8 & 4 & 9 \\ 3 & 2 & 8 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 5 & 4 & 2 \\ 7 & 4 & 5 & 3 \\ 0 & 1 & 4 & 4 \\ 3 & 3 & 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \quad D = \begin{pmatrix} 5 \\ -3 \\ -5 \\ 1 \end{pmatrix},$$

- a) Calculate  $C_1 = A + B$ ,  $C_2 = A - B$ ,  $C_3 = A .* B$ , and  $C_4 = A ./ B$  manually. Show your work.  
b) Use MATLAB to calculate  $C1 = A \times B$  and  $C2 = A/B$ . include a copy of your code and results.  
c) Use MATLAB to find the value of  $X$  when  $D = 2BX$

---

2. Calculate the answer to the following expression:

$$\begin{pmatrix} 5 \\ 12 \\ 9 \end{pmatrix} + \begin{pmatrix} 2 & -1 & 3 \\ 1 & 4 & 5 \\ 0 & 3 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} =$$

---

3. Write the following matrix expression as system of equations:

$$\begin{pmatrix} 6 & -5 & -2 \\ 2 & 4 & 0 \\ 0 & 3 & -1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 7 \end{pmatrix}$$

---

4. Calculate the answer for the following expression:

$$\begin{pmatrix} 5 \\ 12 \\ 9 \end{pmatrix} + \begin{pmatrix} 2 & -1 & 3 \\ 1 & 4 & 5 \\ 2 & 3 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 4 \\ 0 \end{pmatrix} =$$

---

5. Write the following system of equations in a matrix form:

$$\begin{cases} 6x - 5y - 2z = 4 \\ 2x + 4y = 2 \\ 3y - z = 7 \end{cases}$$

---

6. Use MATLAB to plot  $f(t) = \sin(1000t) + \cos(250t)$  and use the `sound()` function to listen to it.

---

7. Use MATLAB's `sin()` and `sound()` function to test your hearing range. A human's range of hearing is between 20Hz - 20kHz. Generate the `sin()` values using 10 samples per period.

---

8. Given the complex number  $(10 + 15j) + \frac{1}{12-10j}$ , write a program to calculate its magnitude, real part and imaginary part. Verify the results by manually calculating its magnitude, real part and imaginary part. Show your work.

*Hint: The `abs()`, `angle()`, `real()` and `imag()` MATLAB functions may be useful.*

---

9. Given the complex number  $\frac{10t+j6}{12-j3}$ , plot its magnitude and phase when  $t = [-10 : 0.1 : 10]$ . Title and label your plots.

---

10. Write a MATLAB function that accepts as input two angles in radians and displays the larger sine value followed by the smaller sine value.

```
function [ ] = sort_sines (x1, x2)
```

---

11. Write a MATLAB function that accepts as input three angles in radians, and displays the cosine values from smallest to the largest, using the following function framework:

```
function [ ] = sort_cosines (x1, x2, x3)

end
```

*Hint: Remember that the `cos(x)` function returns the cosine of angle  $x$ .*

---

12. Write a program that accepts ten real numbers as input and displays the sorted number set from smallest to largest in scientific format.

*Hint: In this program, you are being asked to develop a sorting algorithm, which is a well researched area of computer science. You are encouraged to research existing knowledge in this area.*

---

13. Write a program that stores the 50 consecutive prime numbers starting with 2 in an array and displays the prime numbers in a comma-separated format.

---

14. Use MATLAB to calculate the CPU time required to find the 50 consecutive prime numbers excluding display time. Explore algorithms to reduce the execution time.

---



## Appendix A. Open Source Alternatives to MATLAB

MATLAB is a propriety tool with cost associated with licensing and using it. There are open source solutions available that are free and open accesss to their sources code.

The top two open source alternatives to MATLAB are GNU Octave and SciLab . This section provides a brief overview of each.

### GNU Octave

GNU Octave is the most popular alternative to MATLAB and has had an active development for over three decades. GNU Octave support is available for Widows, IOS and Linux. Octave is also the most compatible alternative to MATLAB. For more information and to download GNU Octave visit [www.gnu.org/software/octave](http://www.gnu.org/software/octave).

### SciLab

Scilab is the next widely used alternative to MATLAB and supports Widows, IOS and Linux. There are some usage differences with MATLAB since 100% compatibility with MATHLAB is not a adevelopment objective. The following table is a sampling of usage differences between MATLAB and SciLab:

<b>MATLAB</b>	<b>SciLab</b>
From File menu open and create new source file. Source file has extension “.m”	From File menu open and create new source file. Source file has extension “.sci”
<i>% when function returns a value</i> function [r]= test(a, b) end	<i>// when function returns a value</i> function r= test(a, b) endfunction
<i>% when function does not returns a value</i> function []= test(a, b) end	<i>// when function does not returns a value</i> function test(a, b) endfunction
Comments begin with %	Comments begin with //
Write the function name in command window to run	Execute Menu, load into scilab and then Write the function name in command window to run.
mod(x,y)	pmodulo (x,y)
fprintf (1,'format string', var1, var2,...)	printf ('format string', var1, var2,...)

For more information and to download Scilab visit [www.SciLab.org](http://www.SciLab.org) .

## Appendix B. Additional Resources

- Additional resources are available at the author's website <http://www.EngrCS.com/>