CHAPTER 1

ENGINEERING PROBLEM SOLVING
Computing Systems: Hardware and Software

• The processor: controls all the parts such as memory devices and inputs/outputs.

• The Arithmetic Logic Unit (ALU): performs the addition/subtraction and other logic operations (OR, XOR, AND, etc.), then stores the result in memory.

• Internal memory: is composed of RAM and ROM.

• External memory: external devices such as external drives or thumb drives.

• Input/output: devices such as printers, CDs and DVDs.
Figure 1.1  Internal organization of a computer.
Computer Software

• Operating Systems: supplies an interface between you (user) and the hardware by providing an environment where you can select and execute software applications.

• Software tools: program written to perform common operations, i.e. word processors, spreadsheet, mathematical computation tools (such as MATLAB).

• Computer languages: first-generation: machine language; second-generation: assembly language; high-level languages or third-generation languages: C, C++, Java; fourth-generation tend to be similar to human language.
Computer Software

Figure 1.2  Software interface to the computer.
Executing a Computer Program

• A C program must be translated into machine language, and a **compiler** is used to perform the translation.

• Next step involves **linking** other machine language statement to an object program.

• Once the program has compiled correctly, then the program can be **executed**. Executing errors, run-time errors must be corrected at this step.
Software Life Cycle

• Definite steps or cycles that are collectively called the software life cycle:

1. Project definition

1. Detailed specification

2. Coding and modular testing

3. Maintenance
Engineering Problem-Solving Methodology

• Problem solving is a key part of engineering courses, as well as courses in computer science, mathematics, physics, and chemistry.

• The process for problem solving we will use has 5 steps:

1. State the problem clearly
2. Describe the input/output information
3. Work the problem by hand for a simple set of data
4. Develop a solution and convert it to a computer program
5. Test the solution with a variety of data
1. State the problem clearly
   Compute the straight-line distance between two points in a plane.

2. Describe the input/output information
   Inputs: Point 1 & Point 2
   Outputs: Distance between points

3. Work the problem by hand for a simple set of data
   Let the points p1 and p2 have the following coordinates:
   p1=(1,5); p2=(4,7)
   distance = \sqrt{(side1)^2+(side2)^2) = \sqrt{3^2+2^2} = 3.61
FIGURE 1.4 Straight-line distance between two points.
4. Algorithm development

The algorithm can be listed as operations that are performed one after another. This outline of steps decomposes the problem into similar steps:

**Decomposition outline**

1. Give values to the two points
2. Compute the lengths of the two sides
3. Compute the distance between the two points
4. Print the distance between the two points

This *decomposition outline* is then converted to C commands
Engineering Problem-Solving Methodology

4. Algorithm development

The algorithm can be listed as operations that are performed one after another. This outline of steps decomposes the problem into similar steps:

**Decomposition outline**

1. Give values to the two points
2. Compute the lengths of the two sides
3. Compute the distance between the two points
4. Print the distance between the two points

This **decomposition outline** is then converted to C commands.
// Purpose: This program computes the distance between two points
// Input(s): two points
// Output(s): distance between points
// Written by: He
// Date: 8/12

#include files

int main(void)
{
    // Declare and initialize variables
    declare variables

    // Compute the sides of right triangle
    compute distance

    // Print distance
    Print distance

    // Exit program
    return 0;
}
5. Testing
The final step in our program is testing the solution.