CPE100
Computer Programming for Engineers
(International Program)

KING MONGKUT'S UNIVERSITY
OF TECHNOLOGY THONBURI

Agenda

- Course Syllabus
- Introduction to C Programming

Introduction

Instructor:	**Assoc. Prof. Dr. Natasha Dejdumrong**		Office: Computer Engineer, Floor 10
Phone No.:	08-5454-5252		E-mail: **natasha.dej@mail.kmutt.ac.th**
Teaching Assistant:	P' Tum		
Lecture Time:	Monday	10:30 — 12:20 AM	Classroom: ONLINE
Lab Time:	Friday	13:30 – 15:20 AM (A,31)	Lab Room: ON-SITE
	Friday	15:30 – 17:20 AM (B,32)	Lab Room: ON-SITE

Zoom Meeting

https://kmutt-ac-th.zoom.us/j/6595353683
Meeting ID: 659 535 3683

Prerequisite: -

Course Description

- Fundamental concepts of programming including data types, conditional execution, iteration, functions, and I/O with programming exercises.
- Software development as a problem-solving activity.
- Techniques for producing correct and robust programs including top-down decomposition, hand simulation and hypothesis-based debugging.
- Weekly laboratory sessions focus on program design and implementation to solve interesting case problems.

KMUTT

Course Objectives

- The objective of this course is to introduce Computer Engineering students to the concepts and practice of programming, using C language constructs as examples.
- By the end of the course, each student should be able to analyze a problem, design a solution to that problem in an abstract form such as pseudocode or flowcharts, code that solution in C, test and debug her solution, and modify that solution in response to changes in the problem specification.
- Students will also learn something about the typical software engineering life cycle since they will be required to revise and extend a previously submitted a project to create a new multi-module software application.

Course Learning Outcomes (CLOs)

- Upon completion of the course, students will be able to:

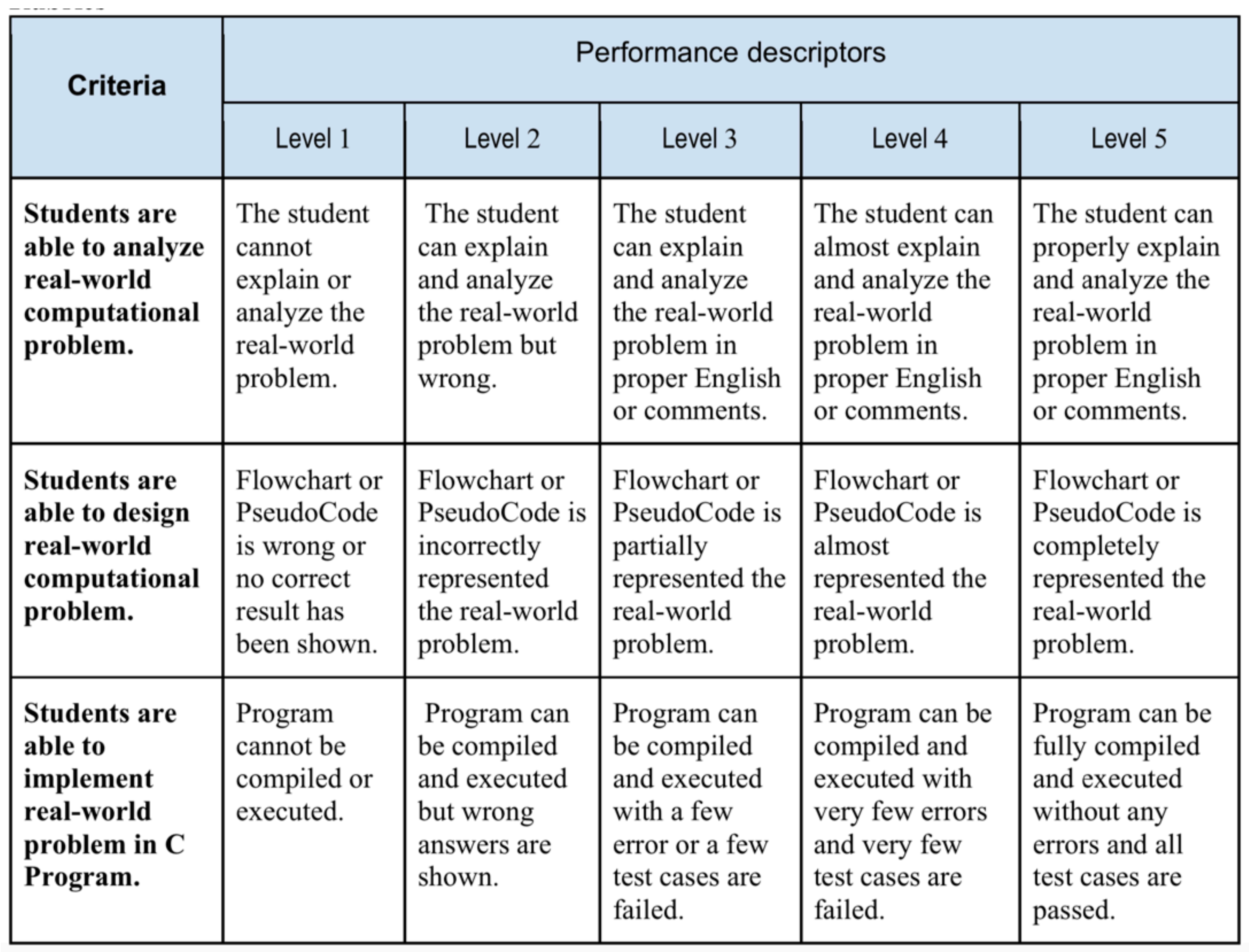
- Design, write and debug a computer program in C that solves a problem as described in a detailed problem specification.

- Create a multi-module software system to solve a problem.

Ultimate Learning Outcomes

- M1: Students are able to analyze, design, and implement basic C programs as described in the problem specification.
- M2: Students are able to analyze, design, and implement intermediate C programs as described in the problem specification.
- M3: Students are able to construct advance C programs by applying the knowledge.

Rubrics



Criteria	Performance descriptors				
	Level 1	Level 2	Level 3	Level 4	Level 5
Students are able to analyze real-world computational problem.	The student cannot explain or analyze the real-world problem.	The student can explain and analyze the real-world problem but wrong.	The student can explain and analyze the real-world problem in proper English or comments.	The student can almost explain and analyze the real-world problem in proper English or comments.	The student can properly explain and analyze the real-world problem in proper English or comments.
Students are able to design real-world computational problem.	Flowchart or PseudoCode is wrong or no correct result has been shown.	Flowchart or PseudoCode is incorrectly represented the real-world problem.	Flowchart or PseudoCode is partially represented the real-world problem.	Flowchart or PseudoCode is almost represented the real-world problem.	Flowchart or PseudoCode is completely represented the real-world problem.
Students are able to implement real-world problem in C Program.	Program cannot be compiled or executed.	Program can be compiled and executed but wrong answers are shown.	Program can be compiled and executed with a few error or a few test cases are failed.	Program can be compiled and executed with very few errors and very few test cases are failed.	Program can be fully compiled and executed without any errors and all test cases are passed.

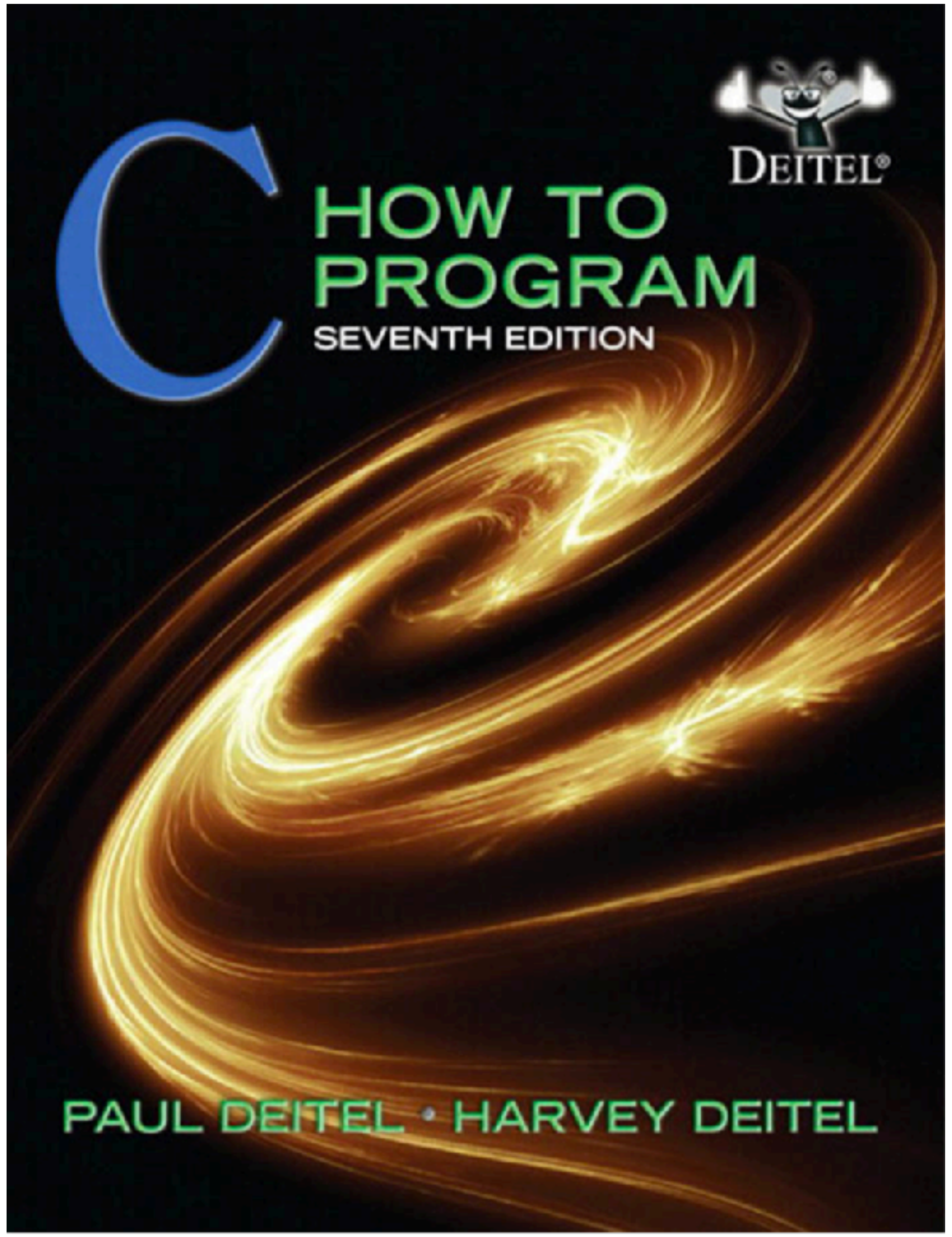
Grading Criteria

Final grades are based on performance indicated by student in-class exercises, lab reports, homework assignments, quizzes, projects, mid-term, and final exam grades. The final grade will be calculated according to the following weights:

Criteria	Weight	Rubric Score	Grade	Weighted Score	
C1	0.25	Level 5	4	1.0	
C2	0.25	Level 4	3	0.75	
C3	0.50	Level 3	2	0.5	
		Level 2	0	0.0	
M1-SUM	0.3			2.25	0.675
M2-SUM	0.5			x	0.5x
M3-SUM	0.2			y	0.2y
SUM		2.25 +	0.5x +	0.2y	

Score range	Grade
3.75 - 4.00	A
3.25 - 3.74	B+
2.75 - 3.24	B
2.25 - 2.74	[✓ C+]
2.00 - 2.24	C
< 2.00	I

Text Book



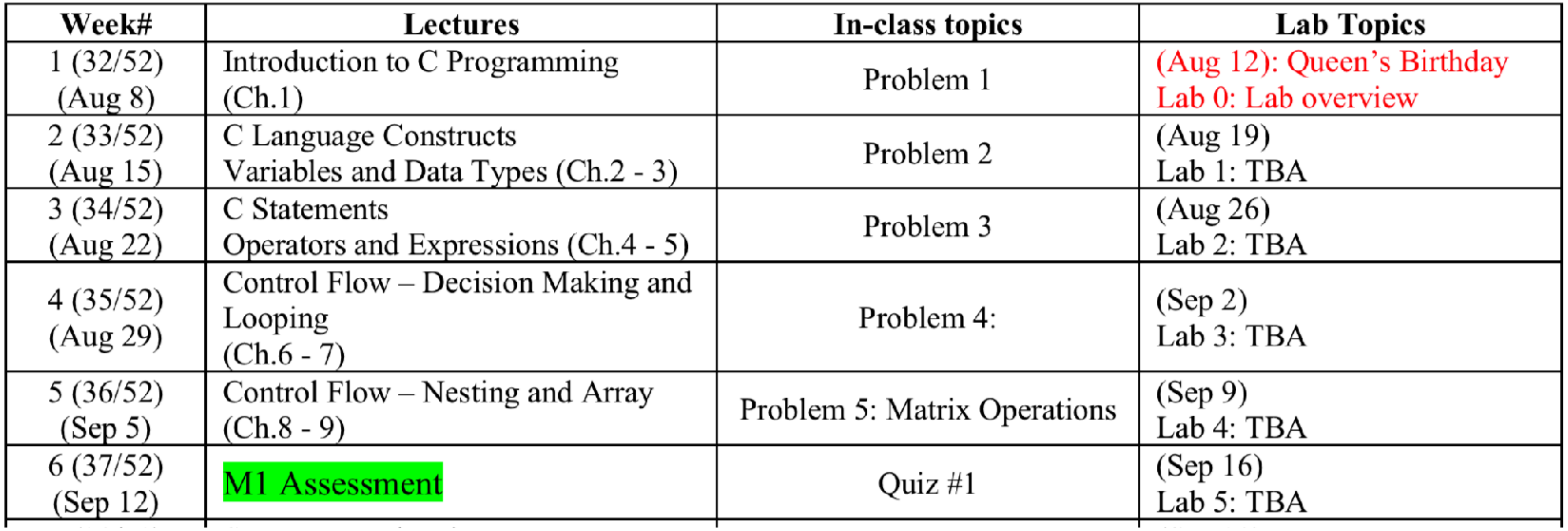
- C How to Program, 7th Edition

  Paul Deitel, Deitel & Associates, Inc.
  Harvey M. Deitel, Deitel & Associates, Inc.
  ©2013.

- Lecture Note

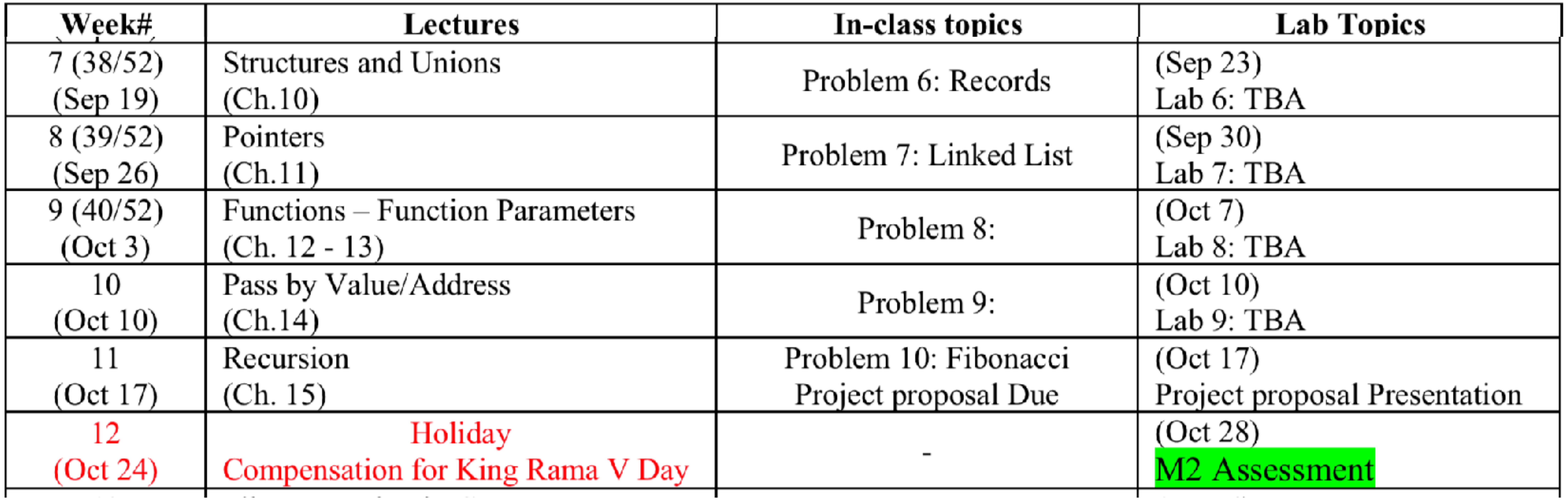
  Computer Programming for Engineers,
  CPE-KMUTT, ©2022.

CPE100 Tentative Schedule



Week#	Lectures	In-class topics	Lab Topics
1 (32/52) (Aug 8)	Introduction to C Programming (Ch.1)	Problem 1	(Aug 12): Queen's Birthday Lab 0: Lab overview
2 (33/52) (Aug 15)	C Language Constructs Variables and Data Types (Ch.2 - 3)	Problem 2	(Aug 19) Lab 1: TBA
3 (34/52) (Aug 22)	C Statements Operators and Expressions (Ch.4 - 5)	Problem 3	(Aug 26) Lab 2: TBA
4 (35/52) (Aug 29)	Control Flow – Decision Making and Looping (Ch.6 - 7)	Problem 4:	(Sep 2) Lab 3: TBA
5 (36/52) (Sep 5)	Control Flow – Nesting and Array (Ch.8 - 9)	Problem 5: Matrix Operations	(Sep 9) Lab 4: TBA
6 (37/52) (Sep 12)	M1 Assessment	Quiz #1	(Sep 16) Lab 5: TBA

CPE100 Tentative Schedule



Week#	Lectures	In-class topics	Lab Topics
7 (38/52) (Sep 19)	Structures and Unions (Ch.10)	Problem 6: Records	(Sep 23) Lab 6: TBA
8 (39/52) (Sep 26)	Pointers (Ch.11)	Problem 7: Linked List	(Sep 30) Lab 7: TBA
9 (40/52) (Oct 3)	Functions – Function Parameters (Ch. 12 - 13)	Problem 8:	(Oct 7) Lab 8: TBA
10 (Oct 10)	Pass by Value/Address (Ch.14)	Problem 9:	(Oct 10) Lab 9: TBA
11 (Oct 17)	Recursion (Ch. 15)	Problem 10: Fibonacci Project proposal Due	(Oct 17) Project proposal Presentation
12 (Oct 24)	Holiday Compensation for King Rama V Day	-	(Oct 28) M2 Assessment

CPE100 Tentative Schedule

Week#	Lectures	In-class topics	Lab Topics
13 (Oct 31)	File Processing in C (Ch.19)	Problem 11	(Nov 4) Lab 10: TBA
14 (Nov 7)	Introduction to Header Files (Ch.16)	Problem 12	(Nov 11) Lab 11: TBA
15 (Nov 14)	C Pre-Processor - Macro (Ch.17-18)	Problem 13	(Nov 18) Lab 12: TBA
16 (Nov 21)	Project Presentation		Project Presentation
17 (Nov 28)	Project Presentation		Project Presentation
Dec 5	Final period, Final		Hands-on Final Exam (TBA)

Introduction to Programming

Programming Languages

- There are three types of programming Languages

1) Machine Language
2) Assembly Languages

3) High-level Languages:
   - Codes similar to everyday English
   - Use mathematical notations
   - translated to machine code by using compilers.
   - C, C++, PASCAL, FORTRAN, BASIC are high-level languages.

Example:

```
c=a+b;
if(a<b)
   printf("a is less than b\n");
else
   printf("a is NOT less than b\n");
```

Introduction to Programming

- In order to communicate instructions to a computer, humans need a language which we call a programming language.
- However, computers nowadays are digital and only understand the language of 0’s and 1’s. Therefore, to instruct a computer to perform some specific task we have to put these 0’s and 1’s in a particular sequence. This kind of a programming language is called Machine Language.
- A set of these instructions is called a Program like a set of dialogs in a spoken language is called a Conversation. Communicating instructions to a digital computer in machine language is difficult, error prone, and non-portable.

Assembly Language

- To overcome some of its drawbacks (specifically difficulty) an assembly language can be used which actually uses mnemonics for these strings of 1’s and 0’s.
- Indeed, assembly language is not understood directly by a computer and hence a translator (which is called assembler) is required to convert these mnemonics into binary strings.
- A translator converts a source program written in some programming language into executable machine code. Machine and Assembly language, both called low-level programming languages, are highly machine dependent, error-prone, difficult to understand and learn.

High-Level Programming Language

- A high-level programming language is machine independent, less error-prone, easy to understand and learn, and so on.
- In this language, the statements (or instructions) are written using English words and a set of familiar mathematical symbols which makes it easier to understand and learn, and is thus less error-prone.
- Furthermore, the language is machine independent and hence can be ported to other machines.

C Programming Language

- C must be noted that all high-level languages also need a translator to convert it into machine language.

- Depending upon whether the translation is made to actual machine code or some intermediate code, the translator is called a Compiler or an Interpreter respectively.

- C programming language uses a compiler to convert the instructions into actual machine code.

- Some researchers also classify C language as a middle level language. The reason behind this is that assembly code can also be mixed with C code.

Why learn C Programming ?

- There are two answers to this question.
- First, C language has been used by programmers for past 30-40 years to develop every kind of utility.
  This means the language is well understood, the issues with the language have been completed eliminated, a lot of the principles used in C show up in a lot of other languages, and so on.
- Second, C combines portability across various computer architectures as provided by any other high-level language while retaining most of the control of the hardware as provided by assembly language.
  It is a stable and mature language whose features are unlikely to disappear for a long time.

History of C Language ?

- In 1965, Bell Telephone Laboratories, General Electric Company and Massachusetts Institute of Technology where working together to develop a new operating system called MULTICS. In 1969, Bell Laboratories ended its participation in the project. However, the participating members of Bell Labs, mainly Ken Thompson and Dennis Ritchie, still had an intention to create such kind of an OS which finally matured into UNIX operating system.

- After it early success in 1970, Ken Thompson set out to implement a FORTRAN compiler for the new system, but instead came up with the language B.
  B Language was influenced by Richard Martins BCPL language which was type-less and interpretive. Hence, both BCPL and B language had some performance drawbacks.

History of C Language

- In 1972, in an effort to add “types” and definition of data structures to B language and use compiler instead of interpreter, Dennis Ritchie came up with a new B language called C language.

- Just like natural languages, programming languages also change. The original specification of the C language (as devised by Dennis Richie) together with close variations is sometimes known as Old-style C or Traditional C.

- However, in 1988 ANSI (American National Standards Institute) published a new specification of the language which has become an international standard which is accepted by all compilers. It is known as ANSI C. ANSI C is mostly a superset (i.e. provides additional functionality) over Old-style C.

Features of C Language

- 1) There are a small, fixed number of keywords, including a full set of control flow primitives. This means there is not much vocabulary to learn.
- 2) C is a powerful, flexible language that provides fast program execution and imposes few constraints on the programmer.
- 3) It allows low level access to information and commands while still retaining the portability and syntax of a high-level language. These qualities make it a useful language for both systems programming and general-purpose programs.
- 4) Another strong point of C is its use of modularity. Sections of code can be stored in libraries for re-use in future programs.

Features of C Language

- 5) There are a large number of arithmetical, relational, bitwise and logical operators.

- 6) All data has a type, but implicit conversions can be performed. User-defined and compound data types are possible.

- 7) Complex functionality such as I/O, string manipulation, and mathematical functions are consistently delegated to library routines.

- 8) C is a Structured Compiled language widely used in the development of system software.

Advantages of C Language

- 1) Easy to Understand: C language is a structured programming language. The program written in C language is easy to understand and modify.

- 2) Middle Level Language: C language has combined features of low-level language (such as assembly language) and some of high-level language. It is, therefore, sometimes C language is also referred to as middle level language. Most of the problems that can be solved using assembly language can also be solved in C language.

- 3) Machine Independent: Program written in C language is machine independent. It means that a program written onone type of Computer system can be executed on another type of Computer.

Advantages of C Language

- 4) Built in Functions: C language has a large number of built in functions. The programmer uses most of these built in functions for writing source program, instead of writing its own code.

- 5) Hardware Control: Like assembly language, the programmer can write programs in C to directly access or control the hardware components of the computer system.

- 6) Easy to learn and use: C language is easy to learn and to write program as compared to low level languages such as assembly language.

Advantages of C Language

- 7) Basis for C++: Today, the most popular programming is C+. C is the basis for C++. The program structure of C is similar to C++. The statements, commands (or functions) and methodologies used in C are also available in C++.
  Thus, learning C is a first step towards learning C++.

- 8) Modularity: C is a structured programming language. The programmer can divide the logic of program into smaller units or modules. These modules can be written and translated independently. Including

Disadvantages of C Language

- 1) C does not have Object-Oriented Programming (OOP) feature.
- 2) There is no runtime checking in C language.
- 3) There is no strict type checking.
- 4) C doesn't have the concept of namespace.
- 5) C imposes few constraints on the programmer. The main area this shows up is in C's lack of type checking.

  This can be a powerful advantage to an experienced programmer but a dangerous disadvantage to a novice.

Introduction to Programming

Structured programming

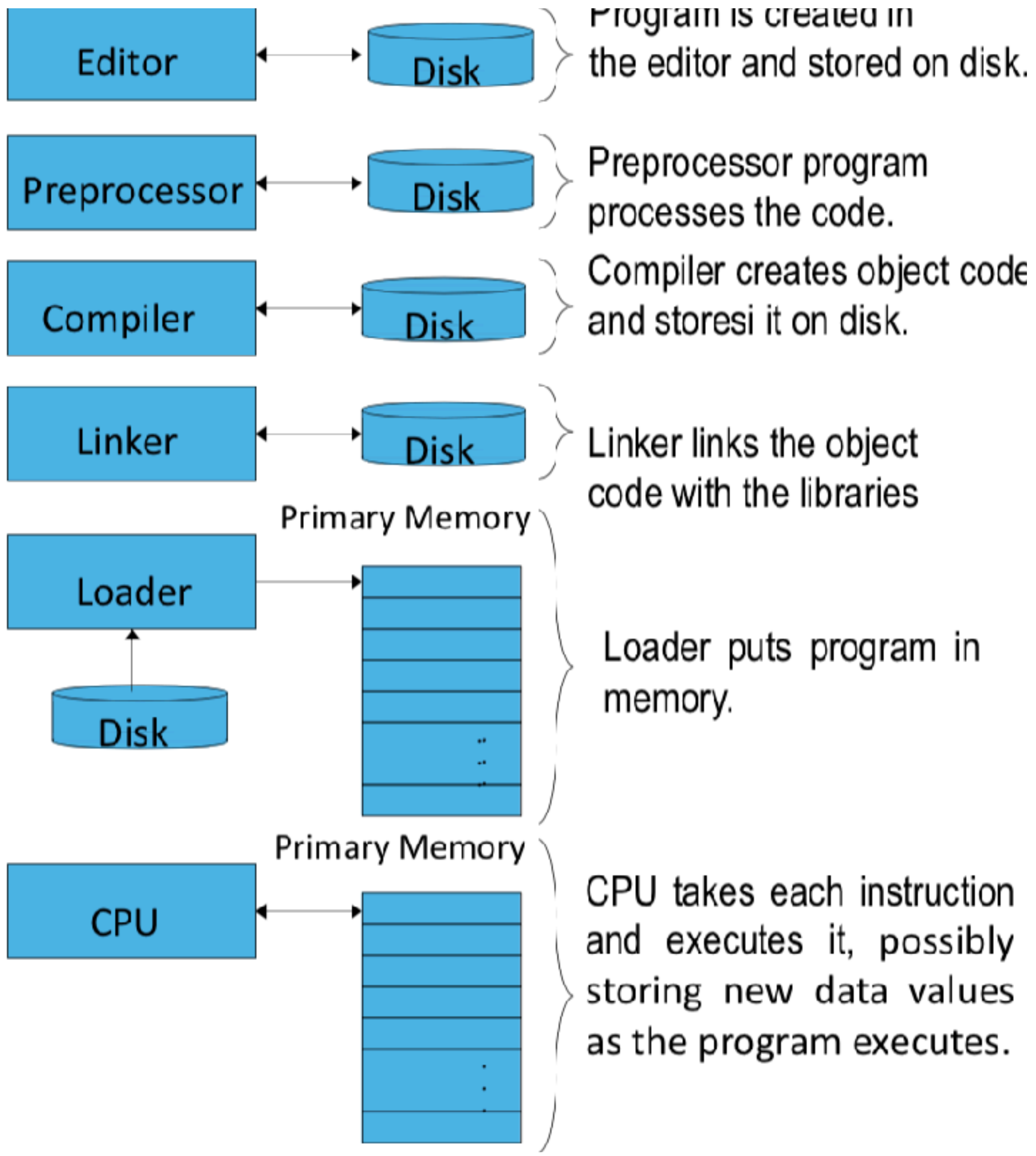
- Disciplined approach to writing programs
  - Using flowcharts (graphical representation)
  - Using pseudocodes or step by step algorithms.
- Clear, easy to test and debug and easy to modify
  - Using functions for efficient programming.

Multitasking

Specifying that many activities run in parallel.

Introduction to Programming

- **Phases of C Programs:**
  1. **Edit**
  2. **Preprocess**
  3. **Compile**
  4. **Link**
  5. **Load**
  6. **Execute**

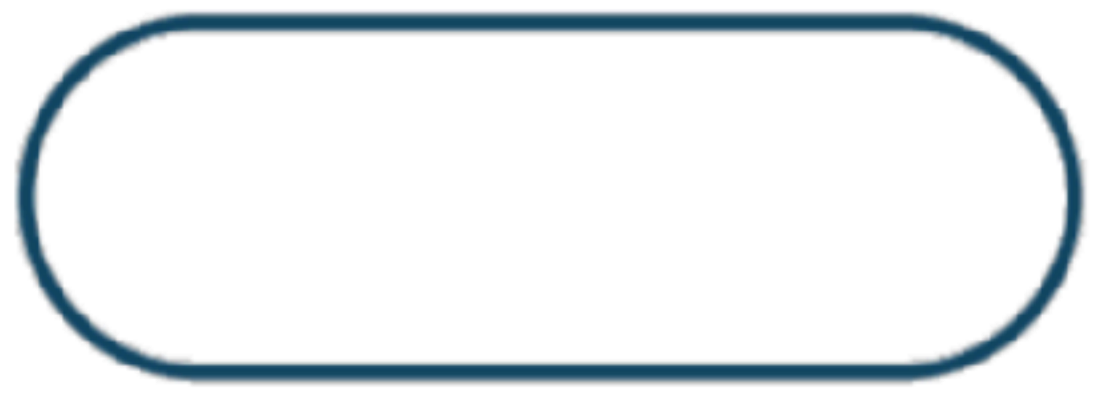


Editor
Disk
Program is created in the editor and stored on disk.
Preprocessor
Disk
Preprocessor program processes the code.
Compiler
Disk
Compiler creates object code and storesi it on disk.
Linker
Disk
Linker links the object code with the libraries
Primary Memory
Loader
Disk
Loader puts program in memory.
Primary Memory
CPU
CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Flowchart

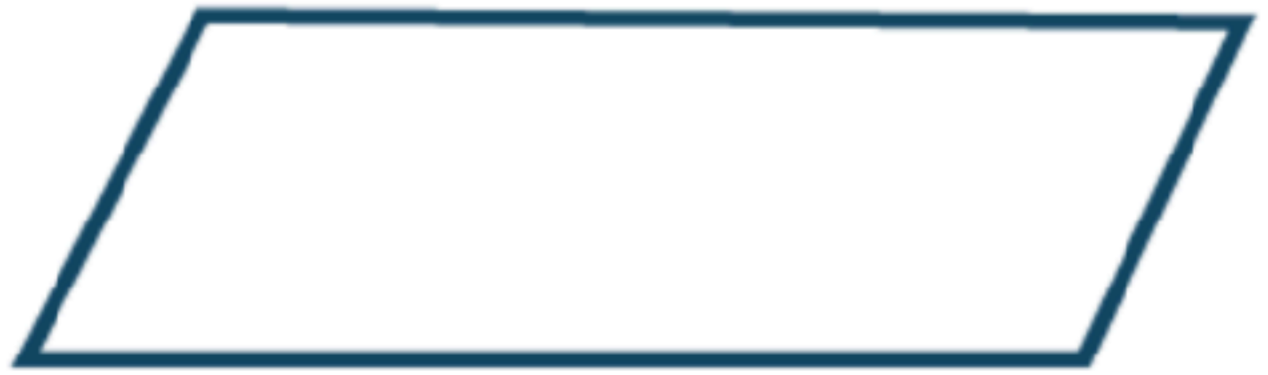
- A diagram represents the workflow or process which helps to solve a task, step by step.
- A flowchart is a pictorial representation depicting the flow of steps in a program, people in an organization, or pages in a presentation.
- Flowchart symbols are universally recognized across multiple disciplines, and will allow an instructional designer to communicate a great deal of information accurately, efficiently, and succinctly.

Flowchart



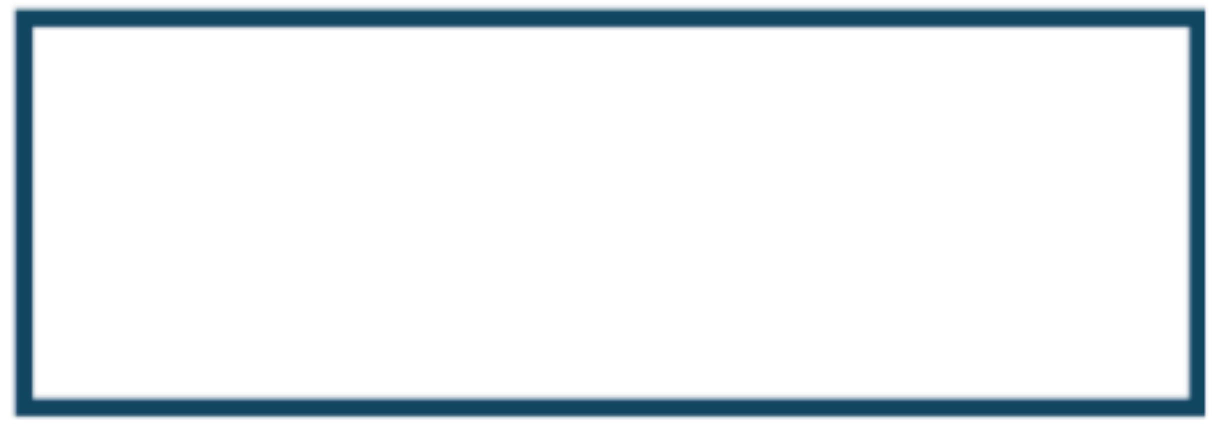
Terminal Symbol: In the flowchart, it is represented with the help of a circle for denoting the start and stop symbol. The symbol given below is used to represent the terminal symbol.

Flowchart



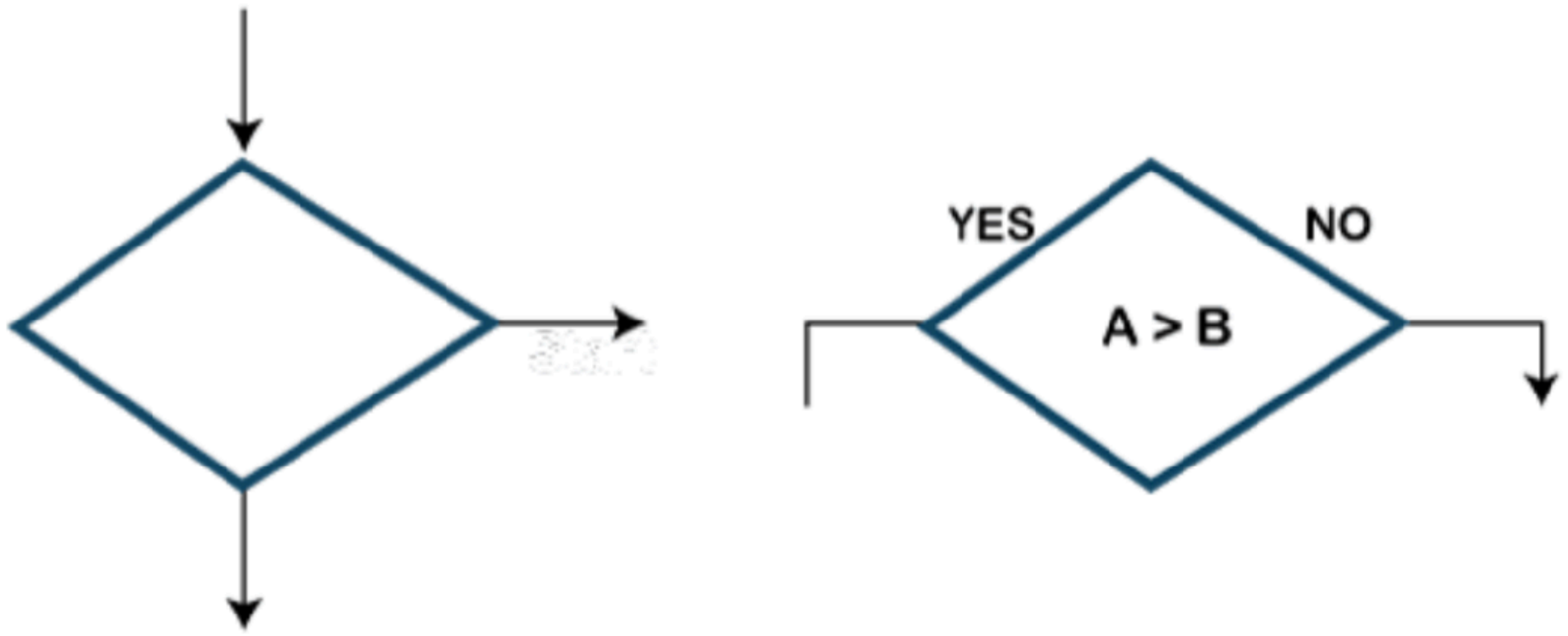
Input/output Symbol: The input symbol is used to represent the input data, and the output symbol is used to display the output operation. The symbol given below is used for representing the Input/output symbol.

Flowchart



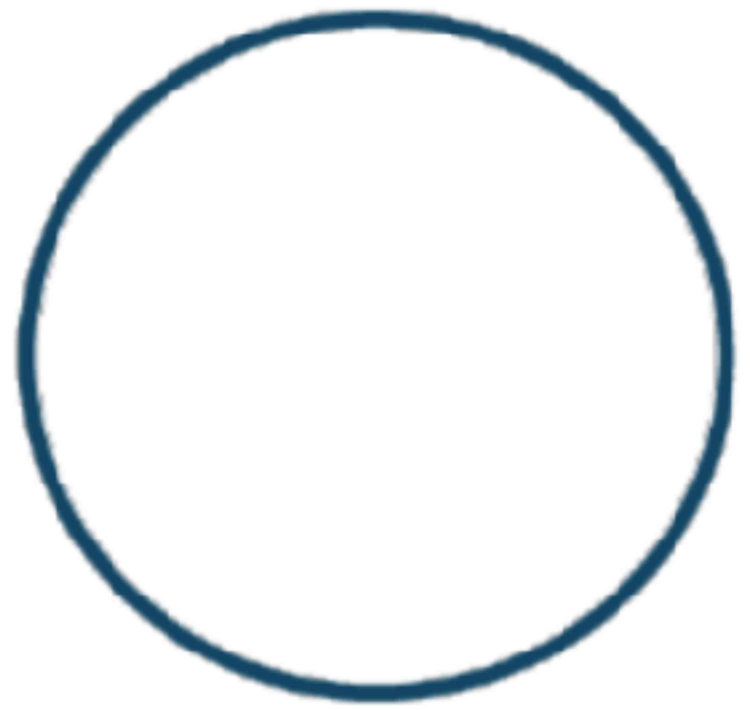
Processing Symbol: It is represented in a flowchart with the help of a rectangle box used to represent the arithmetic and data movement instructions. The symbol given below is used to represent the processing symbol.

Flowchart



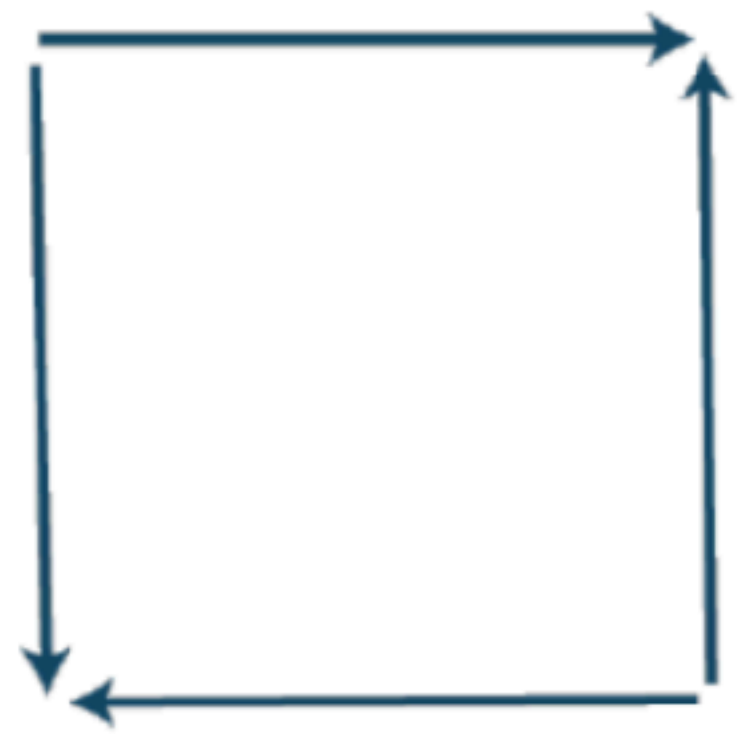
Decision Symbol: Diamond symbol is used for represents decision-making statements. The symbol given below is used to represent the decision symbol.

Flowchart



Connector Symbol: The connector symbol is used if flows discontinued at some point and continued again at another place. The following symbol is the representation of the connector symbol.

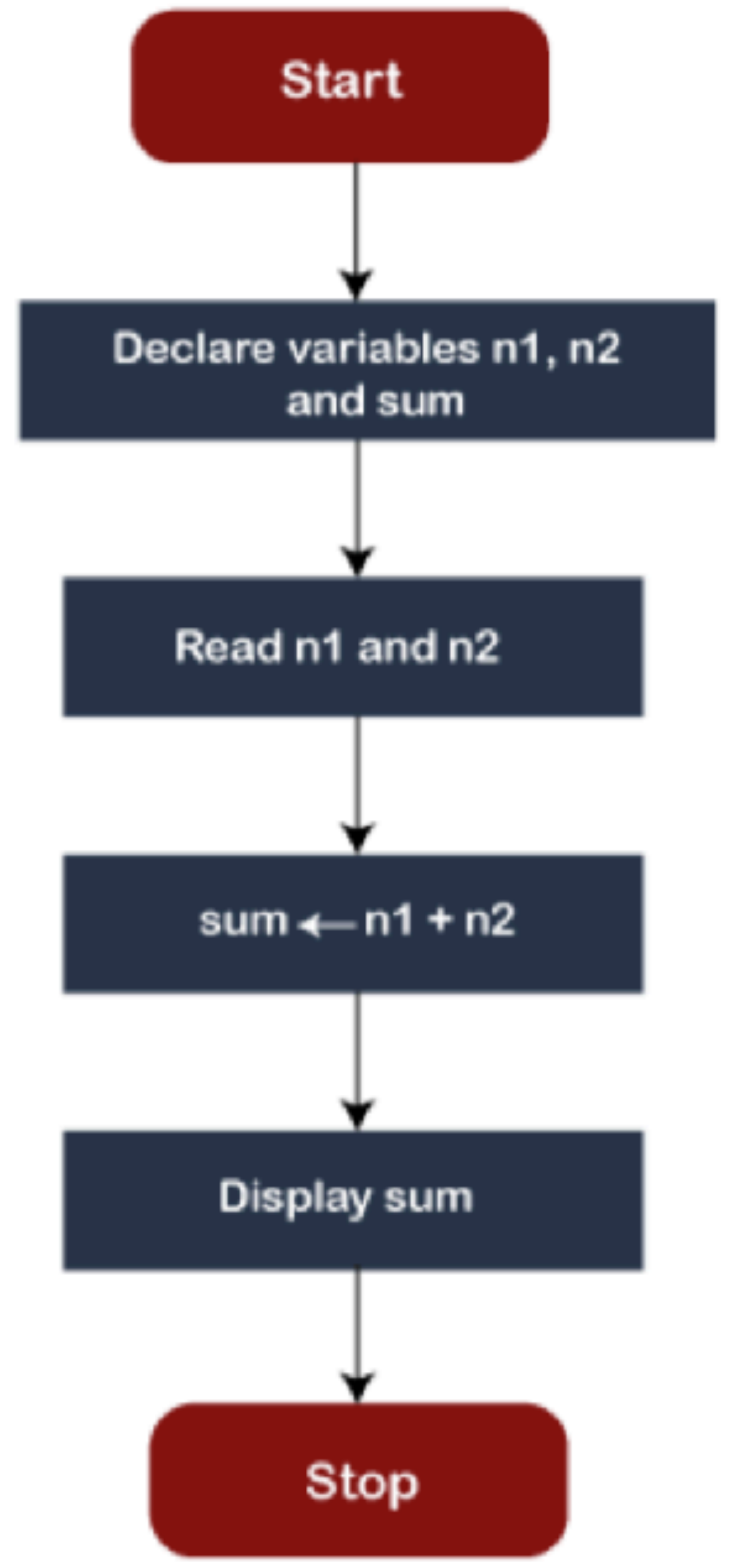
Flowchart



Flow lines: It represents the exact sequence in which instructions are executed. Arrows are used to represent the flow lines in a flowchart. The symbol given below is used for representing the flow lines.

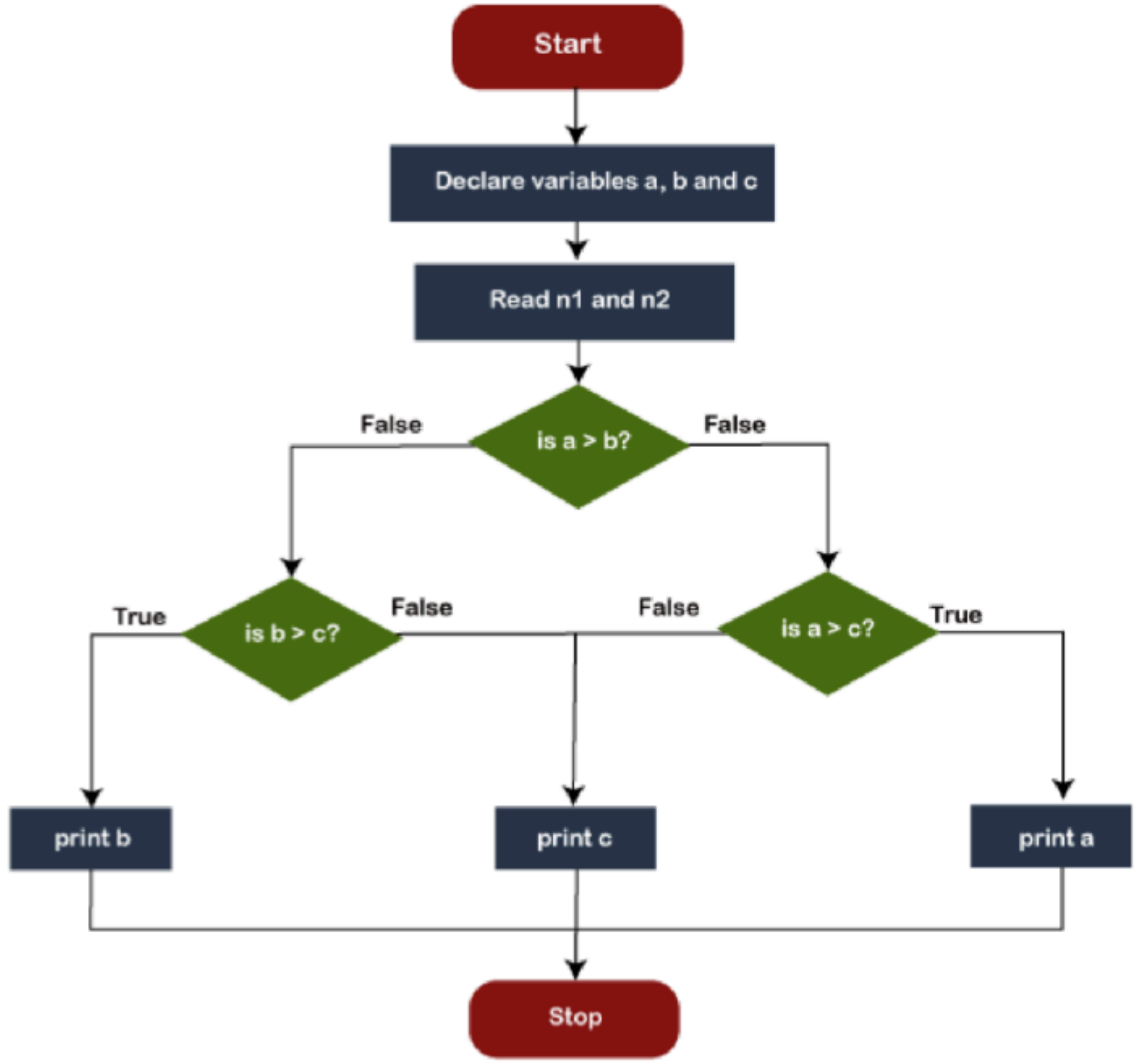
Flowchart

- Example 1: Design a flowchart for adding two numbers entered by the user.



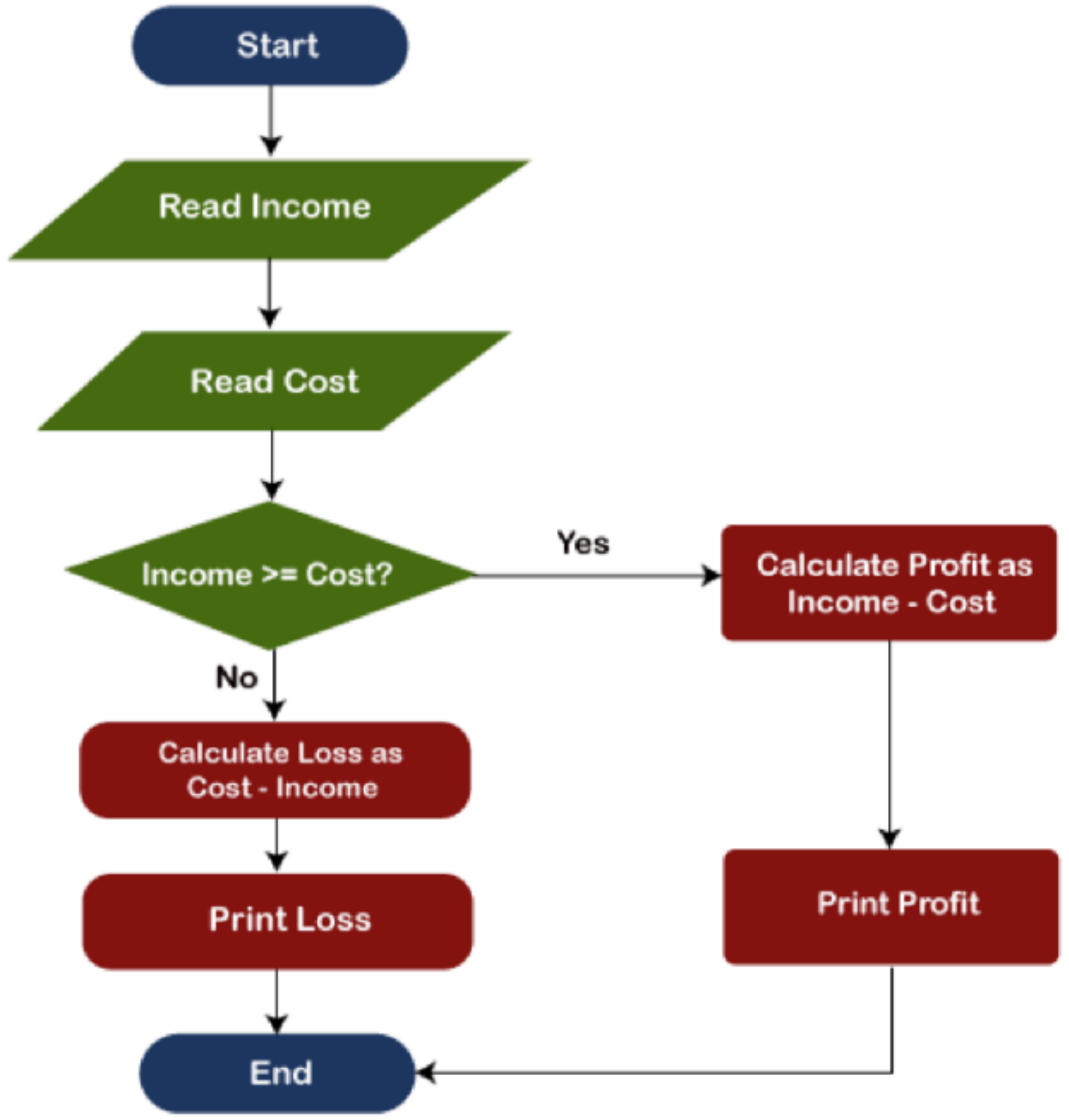
Flowchart

- Example 2: Design a flowchart for finding the largest among three numbers entered by the user.



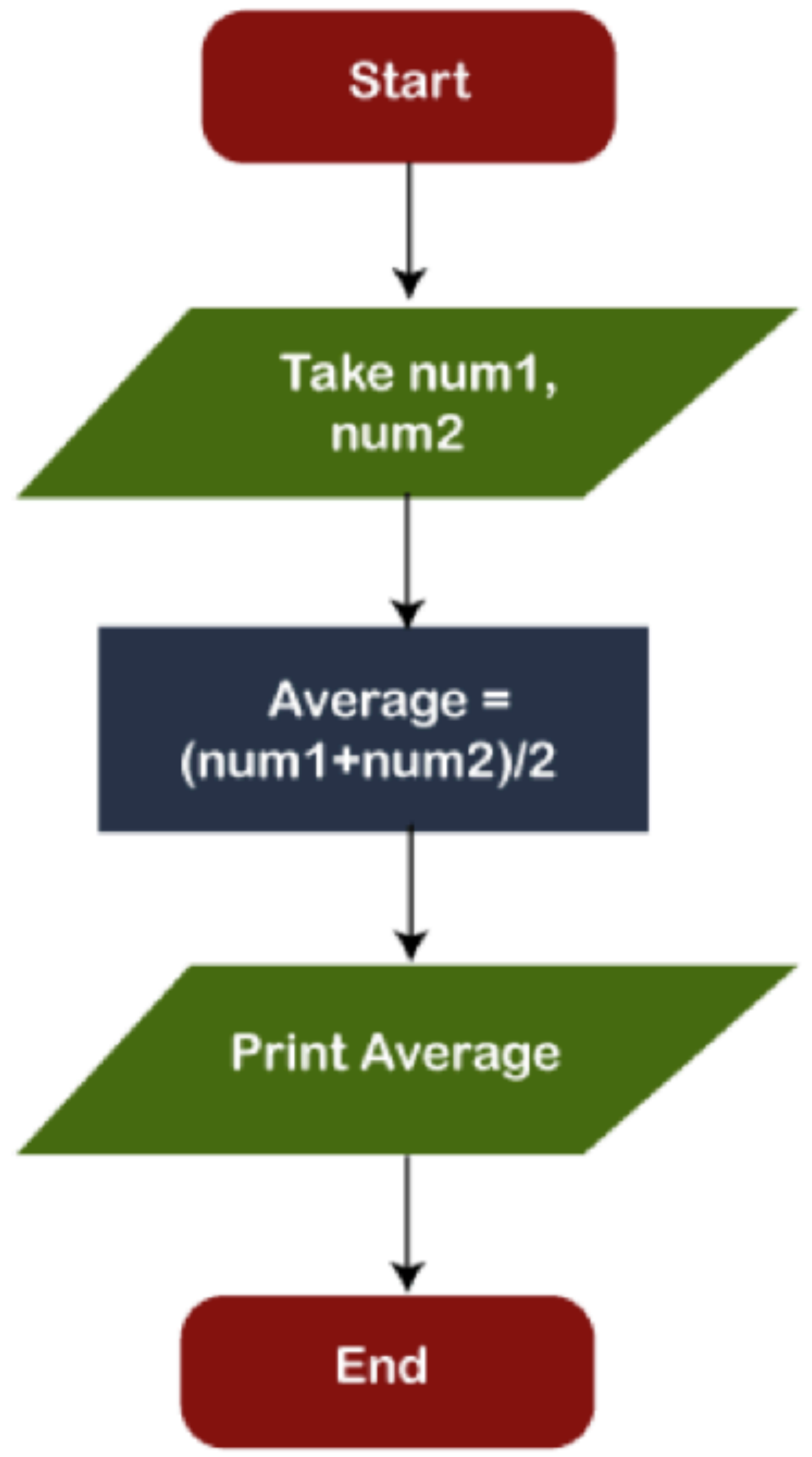
Flowchart

- Example 3: Design a flowchart for calculating the profit and loss according to the value entered by the user.



Flowchart

- Example 4: Draw a flowchart to calculate the average of two numbers.



Pseudo-Code

- Pseudocode is an artificial and informal language that helps programmers develop algorithms.
- Pseudocode is a "text-based" detail (algorithmic) design tool.
- The rules of Pseudocode are reasonably straightforward. All statements showing "dependency" are to be indented.
- These include while, do, for, if, switch.

Pseudo-Code

- Examples below will illustrate this notion.

Example 1:

```
If student's grade is greater than or equal to 60
      Print "passed"
else
      Print "failed"
```

Pseudo-Code

Example 2:

```
Set total to zero
Set grade counter to one
While grade counter is less than or equal to ten
        Input the next grade
        Add the grade into the total
Set the class average to the total divided by ten
Print the class average.
```

Summary

- Programming languages are means to instruct a computer to perform some desired task.
- A program is a set of instructions which are converted into equivalent machine instructions understood by machine.
- All programming languages require a translator to convert the program into machine code.
- C language is a structured compiled programming language which is widely used in the development of application and system software.

Summary

- C language was developed by Dennis Ritchie in order to port UNIX across various platforms.
- C language has rich set of features, innumerous advantages and few disadvantages.
- A flowchart is a pictorial representation depicting the flow of steps in a program, people in an organization, or pages in a presentation.

Questions and Answers