Unit: 2
Programming Logic

Problem Solving:

Problem is defined as the difference between an existing situation and a desired situation, that is, in accordance with calculation; a problem is numerical situation and has complex form. Solution is desired situation and has simplest form. In other words, a problem is the relation between human will and reality. When will and reality do not coincide, the resolution of this gap between reality and will is called problem solving. The process of working through details of a problem to reach a solution is called problem solving. Problem solving may include mathematical or systematic operations and can be a guess of an individual's critical thinking skills. In the context of system analysis and design, a problem is the gap between the current situation and the ideal situation. If a problem is solved by computing using machine called computer, then such process is called Problem Solving using Computer.

Feasibility and Requirement Analysis:

Feasibility analysis is the process to determine whether the solution or system development is feasible according to the organization resources. It is a process which examines the system that is going to be developed and it covers the strength and weakness of the existing system, opportunities as well as threat by the environments and other factors. The resource required to carry the prospects for success. A well designed feasibility study should provide a historical background of the system. During the study, the problem definition is solved and all aspects of problem to be included in the system are determined. Size of project, cost and benefits are also estimated with greater accuracy. A feasibility study is conducted by highly skilled personnel called system analyst, to assist decision maker in deterring whether or not to implement a particular system. It is an important step in system development process. The result of feasibility study is simply a report, which is a formal document detailing the nature and scope of the proposed solution.

Project managers use feasibility studies to determine potential positive and negative outcomes of a project before investing a considerable amount of time and money into it. Feasibility studies contain comprehensive, detailed information about our business structure, our products and services, the market, the logistics of how we will deliver a product or service, and the resources we need to make the business run efficiently.

A feasibility analysis of a proposed system is undertaken in following headings (TEBLOS):

- Technical feasibility analysis
- Economical feasibility analysis
- Behavioral feasibility analysis
- Legal feasibility analysis
- Operational feasibility analysis
- Scheduled feasibility analysis

A. Technical feasibility analysis

It focuses on the availability of technical resources and how it can be integrated within the organization. It helps organizations to determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility analysis consist evaluation of hardware, software, human-ware, and other technology requirements of the proposed system. In technical feasibility the following issues are taking into consideration

- Whether the required technology is available or not?
- Whether the required resources (software and human-ware) is available or not?

B. Economical feasibility analysis

Economical feasibility analysis is most frequently used method for evaluating effectiveness of new system. The purpose of assessing economical feasibility is to identify the financial benefits and cost associated with the development of the system. It is also known as cost-benefit analysis (C/B analysis). It is used to determine the benefits and saving that are expected from the new system and compares them with cost, the system is designed and implemented otherwise the system is rejected.

C. Behavioral feasibility analysis

Behavioral feasibility is the analysis of behavior of the candidate system. It includes how strong the reaction of staff will be towards the development of new system that involves computer's use in their daily work. So resistant to change is identified. People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development of a computerized system. It is common knowledge that computer installations have something to do with turnover, transfers, retraining, and changes in employee job status. Therefore, it is understandable that the introduction of a candidate system requires special effort to educate, sell, and train the staff on new ways of conducting business. It also analyze the change in behavior of the users due to the change of system.

D. Legal feasibility analysis

This legal feasibility is performed to analyze the conflicts and legal requirements, data protection acts or social media laws that arise between system under consideration and the organization. It also considers any legal aspect from the society and government rules. If the proposed system is legally feasible, the new system is adopted otherwise the system is rejected. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

E. Operational feasibility analysis

This analysis involves undertaking a study to analyze and determine whether - and how well - the organization's needs can be met by completing the project. It also analyzes how a project plan satisfies the requirements identified in the requirements analysis phase of system development. It is mainly concerned with issue like whether the system will be used if it's developed and implemented whether there will be resistance from users etc. will affect the possible benefits from the system. The operational aspects are really a human related problem. For better operation of the proposed system, training should be given to the users who are performing the function of new system.

F. Scheduled feasibility analysis

It is the most important for project success; after all, a project or system will fail if not completed on scheduled time. Typically schedule feasibility means estimating how long the system will take to develop? It is a measure of how reasonable the project time table is? If the system cannot deliver in the scheduled time then it can be outdated and the goodwill of the system developing institution will be decreased.

Design (Flowchart and Algorithm)

Algorithm:

The term algorithm may be formally defined as a sequence of instructions designed in such a way that if the instructions are executed in the specified sequence, the desired result will be obtained. An algorithm is set of instruction that computer follows generally complete a task. It is a description of series of steps used to solve a specific problem. An algorithm is: *An effective procedure for solving a problem in a finite number of steps.* The algorithm is part of the blueprint or plan for the computer program.

Features of algorithm:

- It should be simple
- It should be clear with no uncertainty
- It should lead unique solution of the problem.
- It should involve finite number of steps.
- It should have capability to handle some unexpected situations which may arise during the solution of problem.

Different patterns of algorithms

a) **Sequential** : In this different steps occur in a sequence

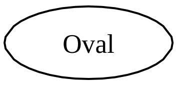
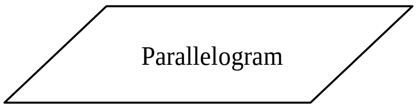
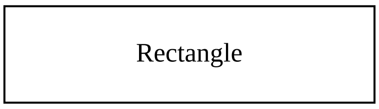
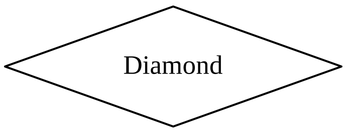
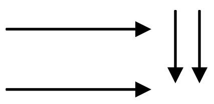
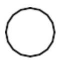
b) **Conditional:** In this different steps are executed based on a condition (whether true/false). In programming languages, a conditional pattern is implemented using decision making statements.

c) **Iteration:** In this a task (one or more steps) is repeated more than once. In programming languages, an iteration pattern is implemented using loops. An iteration construct is also known as repetitive construct

Flowchart:

A flowchart is a pictorial representation of an algorithm that uses boxes of different shapes to denote different types of instructions. The actual instructions are written within these boxes using clear and concise statements. These boxes are connected by solid lines having arrow marks to indicate the flow of operation, that is, the exact sequence in which the instructions are to be executed. It is also known as "flow diagram." It shows steps as boxes or levels that connect by lines or arrows. Each box contains a thought or action, and the arrows provide flow and direction. Flowcharts provide a streamlined way to organize, analyze, design and document.

The various symbols are used to write different instructions/operations.

Symbol	Meanings
	Terminal box: start and stop
	Input / Output box: Accept and display result.
	Processing and assignments: It is used to process and assignment values.
	Decision box: Take decision in some conditions
Arrows 	Flow lines: connect to carious component of flowchart and show the logical sequence of step.
Small circle 	Connector: A long flowchart which is in multiple pages or different part of flowchart in a single page are connected to each other using this symbol. It must be filled up by a label. Exit connector , entry connector

Advantages of using flowcharts

- Flowcharts are better way of communicating the logic of a system.
- With the help of flowchart, problem can be analyzed in more effective way.
- It serves as a good program documentation, which is needed for various purposes.
- The flowcharts act as a guide or blueprint during the systems analysis and program development phase.
- It helps in debugging process.
- It helps the programmer to put efforts more efficiently on the maintenance of a program.

Disadvantages:

- Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
- If alterations are required the flowchart may require re-drawing completely.
- As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.

Program Coding (Execution, Translator)

Coding the programs means translating the solution of the problem which is specified either by flowchart or algorithm into a programming language. The language chosen depends on what the program expected to do and what facilities are available to the programmer. The programming activity continues with the coding of the logic, describing in the flowchart or algorithm into the instructions of particular programming language.

Language processor (Translator):

Language processor or translator is a system program that is used to translate any other programming languages into its equivalent machine level codes. There are three types of language processors. They are: assembler, compiler and interpreter.

Comparison between Compiler and Interpreter

Characteristics	Compiler	Interpreter
Translation method	It translates high level programming's source code into object codes as a whole.	It translates the statements of the source code one by one and execute immediately.
Creation of object files	Creates and stores permanently	Doesn’t create and doesn’t store.
Program execution speed	Very fast.	Slow.
Translation required	Translator program is not required to translate the program each time we want to run the program, unless its modification.	Translator program is required to translate the program each time we want to run the program.
Debugging process	It does not make easier to correct the mistakes in the source code	It makes easier to correct the mistakes in the source code
Listing of errors	created after the compilation process	Stops translating after the first error.
Number of high level programs	Most of the programming languages where storage is high.	A few programming languages have Interpreter program, where storage capacity is very low
Example	C, C++, C#, etc	Qbasic, LISP, et

Assembler:

A language processor that is used to convert low level languages into its equivalent machine codes is known as assembler. There are two types of assemblers: self assembler and cross assembler.

Testing and Debugging

Testing:

Testing is the process of reviewing and executing a program with the intent of detecting errors. Testing can be done manually and computer based testing. It is a process of examining software whether the software is functioning according to the specification or the software fulfills the objectives of developments. In other words it is a procedure to establish the quality performance, presence etc of software.

The new system should be tested for system compatibility and operational efficiency before it is used. It involves running the program to process input test data and compare the produced results with the known correct results. If the result doesn't matches, then it is assumed that the program contains one or more logical errors. If a program runs successfully with the test data and produces correct result, it is normally released for used. However, error my still remain in the program. Hence, it is impossible to certify that such system re free of all logical errors. After extensive in house testing of commercial software is completed, a beta version of the software is released for selected set of users. The selected users uses the beta version and reports errors (if any) to the software vendor the programmer collects all the feedback from the users and starts correcting those errors and such process is repeated until the complete satisfaction of the user. This repeated testing is known as regression testing. Finally the product is lunched into the market for commercial purpose.

Generally programmers commit three types of errors. They are: Syntax errors, logic errors and run time error.

a) Syntax error

Syntax is a grammatical rule for writing a programming statement. Syntactical error are those errors which are occurred when the rules or the syntax of any programming language are not followed. Such programming errors are generally involved when incorrect punctuation, incorrect word sequencing, undefined terms, and misuse of terms etc. a program cannot be successfully compiled and executed until all its syntactical errors have been corrected. It is very easy to debug because the compiler itself detects such syntax error and also describes the reason of errors. Different kinds of language processors are used to identify syntactical errors.

b) Syntactical errors

The error caused by the violation of semantics of programming languages is called logical error or semantical error. It is an error in planning the programs logic such error causes the program to produce incorrect output that means a program which is free from syntactical error but has one or more logical error, such errors are successfully compiled and execute so, such errors are hard to identify as compared to syntactical error.

If we are writing a program to read nay two number and adding them but instead of + symbol we write – symbol, the program will successfully compiled and produce the result such result is not expected by the user. Such type of error is known as logical error.

c) Runtime error

An error that occurs during the running time of software is called run time error. Such types of error are occurred due the problem of system or mishandling of the software, infinite loop statement, device errors, etc. Such types of errors can be handled by programmers by different error handling mechanisms. The computer will print the error message .Some of runtime errors are:

- Divide by zero
- Null pointer assignment
- Data over flow

There are two types of testing strategies: top down and bottom up approach.

Top down Approach

In this method the testing is done by beginning from the main module and shifting toward the low level modules. When the main module is tested, it is broken down into some sub-modules, such sub0modules re tested and again they are broken down not other sub-modules. This process is repeated until the last module is tested,

Bottom up approach

In this method, the lower level module is tested at first and then it is shifted towards upper level modules. It is just opposite of top-down approach. After all the lowest level modules are tested, they are integrated into some common module and again tested as complete system.

Both testing strategies can be divided into two types:

a) Black box testing

In this technique the internal coding of the programs are tested. It is also known as functional testing because it test al the functional parts of the codes. It is called black box because the test cases are completely hidden for the general users. It generally test the loop and their functions, controls structure and theirs functions, array and their boundaries, etc.

b) White box testing

In this technique the structural part of the program is tested. It is used to check whether all the possible paths in the loops or the control flow re corrected or not it is also known as glass box testing because the text cases are totally visible to the general users.

Debugging:

A program is not 100% correct throughout its life span. There might be some mistakes due to violation of rules of programming language and it is known as errors or bugs. The process of finding bugs and correcting them is known as ***Debugging***. One simple method of debugging is to place print statements throughout the program to display the values of variables. It displays the dynamics of a program and allows us to examine and compare the information at various points. Once the location of an error is identified and the error is corrected, the debugging statements may be removed.

Syntactical errors and logical errors are collectively known as bugs and the process of eliminating these errors is known as debugging. In other words a debugging is a process of locating errors and fixing them. Almost all the programming language has its own language processor which is designed to detect syntactical errors of any programming statements. If any error is detected, the language processor makes a list of syntactical errors and gives hints as the nature of these errors. Hence a programmer can easily eliminate such errors in less time and effort.

A computer doesn’t produce any error message for logical errors in a program hence, its tough job to detect logical errors. However following methods may be used to locate and correct logical errors.

a) Debugger

A debugger is a software tool which assists the programmer in the following the program execution step by step by allowing, displaying intermediate calculation and other values, whenever required. This most commonly used approach, but such tool is not available in all programming language.

b) Memory dump

This method is used when the program hangs up during a test run. A printout of the contents of main memory and registers is carried out at the time of hang and such printout is called memory dump through which a programmer can locate errors and take corrective actions.

c) Hand simulation of program code

In this method a printout of complete source code of the program is made and performed execution manually with the test data, which produced incorrect, results this method is suitable only for simple and small programs.

d) Putting print statements on the source program

In this method several print statements are inserted at appropriate locations in the program, so that the values of different variable can be displayed to indicate intermediate calculations result. The program is re complied and execute with the results. Once the error have been found and corrected, these print statements are removed from the program.

Difference between testing and debugging

Testing	Debugging
It is a process of validating the correctness of the program.	It is process of fixing and eliminating errors in a program
Its objective is to determine whether the program meets its design and specification	Its objective is to detect exactly cause of errors and remove known errors in the program.
Testing is complete when all desired verification is completed	Normally debugging is completed when all the known errors have been fixed. However, it is a continuous process, which may be restarted whenever new error is found in the program.
	It is a re-active process which cannot be planned in advance.

Implementation:

Implementation refers to the final process of moving the solution from development status to production status. This process is often called deployment, go-live, rollout or installation.

Evaluation and Maintenance of Programs:

Once the program is thoroughly tested and all known errors have been removed. The program also with required hardware is installed for use by the intended users. At this stage the old system (which may be manual program or inefficient program) is phase out and new program is phased in.

Once the new program is implemented and put to operation, it is necessary to evaluate the program to verify whether it is meeting its objectives or not. These objectives of the program are clearly defined in problem identification and requirement analysis steps. While evaluating a system the following points are normally considered.

- ***Performance Evaluation:*** The performance of the new program system is evaluated and compared with the old system. In general the new system should be at least as efficient of the old system on performance.

- ***Cost Analysis:*** It should be analyzed whether the cost estimate done for the various steps of the program during initial step match with the actual cost occurred in each step.

- ***Time Analysis***: It should be analyzed whether the time estimate done for the various steps of the program during the initial steps matches the actual time taken in each step.

- ***User Satisfaction:*** It should be found whether the users are satisfied with the new system? How useful is the program for them, they receive output in time to take necessary actions? All the questions are analyzed to evaluate the level of user satisfaction.

- ***Failure Rate:*** The quality of a program also depends on its failure rate. A program which frequently fails can't meet its objectives successfully.

- ***Ease of Modification:*** Soon or later all the programs need to be modify to meet new requirements. Therefore the program should be designed and developed such that it is easy to modify

Documentation:

A document is a paper artifact containing information in the form of ink marks. Documentation is the process of collecting, organizing, storing and maintaining a complete historical record of program and other documents used or prepared during the different phase of the life cycle of software. It is an in-going process, which start in the study phase of the software development and continues, until its implementation and operation phase. It is the process, which never ends throughout the life of the software.

Document is a collection of past historical record of any object. The process of writing the document by explaining the system details of various stage of system development process which helps to analyze a new proposed system or the existing old system is known as documentation. Generally documentation process is performed by a system analyst for each and every stage of a development process. For large software projects, it is usually the case that documentation starts being generated well before the development process begins. A proposal to develop the system may be produced in response to a request for tenders by an external client or in response to other business strategy documents. For some types of system, a comprehensive requirements document may be produced which defines the features required and expected behavior of the system. During the development process itself, all sorts of different documents such as: project plans, design specifications, test plans etc may be produced.
The documents associated with a software project and the system being developed has a number of associated requirements:

- They should act as a communication medium between members of the development team.
- They should be a system information repository to be used by maintenance engineers.
- They should provide information for management to help them plan, budget and schedule the software development process.
- Some of the documents should tell users how to use and administer the system.

Satisfying these requirements requires different types of document from informal working documents through to professionally produced user manuals. Software engineers are usually responsible for producing most of this documentation although professional technical writers may assist with the final polishing of externally released information.

The documentation should include:

1. ***Definition of the problem:***
   It includes why the software is developed? What are the objectives of the system? Who requested for it and who approve it. These questions must be answered in this section of documentation.

2. ***Description of the software system:***
   The system and it's complete structure including its scope, it's functions and all type of input data and required output should be clearly specify.

3. ***Description of the program:***
   Documentation must include complete logic of the program using flowcharts or algorithms, test data and expected results etc.

4. ***Description of operator instructions:***
   Documentation must include a complete instructions to it's users about how to load and unload the program, starting, running and terminating the program.

5. ***Description of program controls:***
   A description of controlling the program when unexpected event occurs must be specified in the documentation.

Forms of documentation:
There are three commonly used forms of documentation: comments, system manual and user manual, which are necessary to properly and completely document software.

1. **Comments**
   - Comments are natural language statements put within a program to assist anyone reading the source program, listing in understanding the logic of the program.
   - They do not contain any program logic and are ignored by language processor.
   - All high level language provides the facility to write comments along with the source code of a program.

2. **System manual:**
   System documentation describes the systems' functions and how they are implemented. Most system documentation is prepared during the system analysis and system design phases. This documentation consists of: data dictionary entries, data flow diagrams, screen layouts, source documents, etc. Good software must be supposed with a standard system manual, which contains following information's
   - A statement of problem clearly defining the objectives of developing software.
   - A statement or description of the software specifying the scope of the problem, the environment in which it works, its limitations, its input data requirements and type of output required.
   - Detail system flowcharts and program flowcharts.
   - Specimen of all input forms and printed outputs.
   - Specification of all input and output media required for the operation of various programs.

3. **User manual:**
   A good software package should be supported with a good user manual to ensure the smooth running of the package. It must contain following information's:
   - Set up and operational details of each programs.
   - Loading and unloading procedures.
   - Starting, running and terminating procedure.
   - A description and example of any control statements.
   - List of error conditions with explanations
   - List of program to be executed before and after execution of each program.

Requirements of documentation
Proper documentation of software is necessary due to:

a) It solves the problem of indispensability of and individual for an organization. Even a person, who has designed or programmed the software, leaves the organization, the documents knowledge remains with the organization which can be used for continuity of the software.
b) It makes software easier to modify and maintain in the future. It is easier to understand the logic of a program from the document records, rather than its code. System flowchart, program flowcharts and comments are very useful in this regard.
c) It helps restarting a software project, which was postponed due to some reason. It avoids duplication of work and saves a lot of time and effort.

Good documentation:
Documentation is an essential activity for the development of software. A good documentation should have following features:

1. Good documentation starts with the statement of the problem, why the program was created? What the program can do and how it is done?
2. It breaks the entire program into modules and explains the function of every module. It gives and how it implements it's task. It also gives the logic of the program using flowchart, pseudo-code and even complete coding of the program.
3. Each modification must be fully documented with modified flowcharts and program code.
4. System administrator or project leader should ensure that every modification has been documented properly.
5. The complete documentation must be stored in such a media form where it can be easily accessed.
6. Suitable comments or remarks should be given wherever necessary.

Importance of documentation
Documentation is an important activity of software development process. The purpose of documentation is:
Communication:
Documentation plays an important role for communication among different group of people involved in a software development. For example, the analyst team discovers all user requirements; prepare a document and handover it to the designer team for next phase of software development. So document is used for communication among different stages of software development. It is also used for proper communication among the people of same stage of software development.

Quality control:
Software quality has to be checked at regular interval throughout it's production. Documentation is part of "exit criteria" (or final output) from each stage of software development process because the quality of one reflects the quality of next stage. For example, after completing the analysis stage the final output of this stage must be checked to maintain the quality of the next stage and so on.

Reference:
Documentation is a reference source for the lifetime maintenance of the software program. Since the user requirements may change and we have to maintain the software. Document works as a reference for the member of the software maintenance team.

Examples of Algorithms in Programming

a. Write an algorithm to add two numbers entered by user.

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
          sum←num1+num2
```

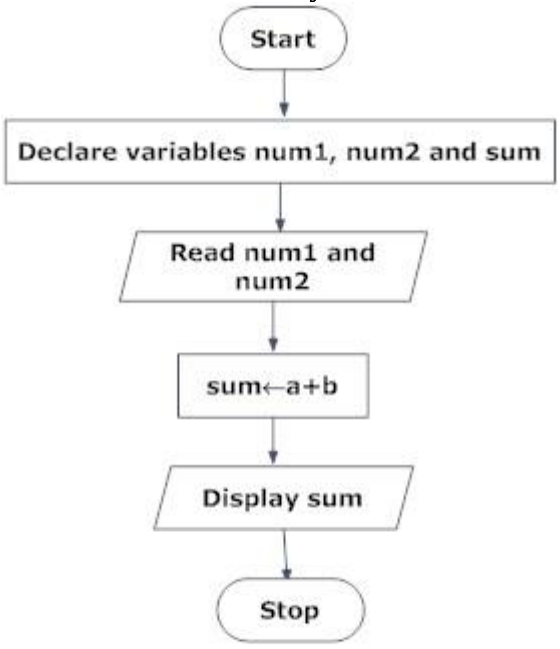
```
Step 5: Display sum
Step 6: Stop
```

b. Write an algorithm to find the largest among three different numbers entered by user.

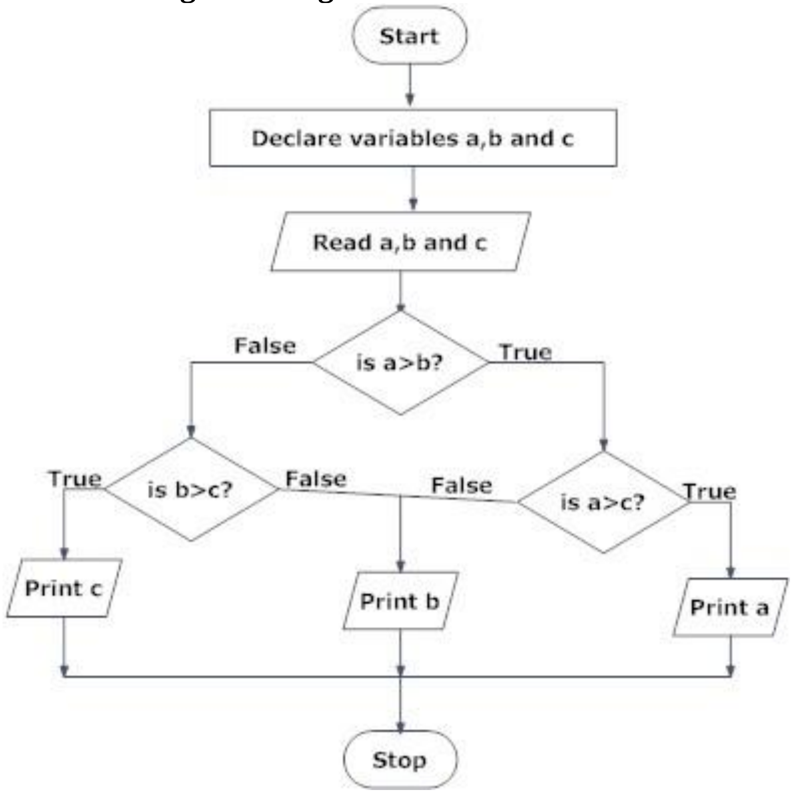
```
Step 1: Start
Step 2: Declare variables a, b and c.
Step 3: Read variables a, b and c.
Step 4: If a>b
           If a>c
              Display a is the largest number.
           Else
              Display c is the largest number.
        Else
           If b>c
              Display b is the largest number.
           Else
              Display c is the greatest number.
Step 5: Stop
```

Examples of flowcharts in programming

a. Draw a flowchart to add two numbers entered by user.



b. **Draw flowchart to find the largest among three different numbers entered by user.**


Start
Declare variables a,b and c
Read a,b and c
False
is a>b?
True
True
is b>c?
False
False
is a>c?
True
Print c
Print b
Print a
Stop