New
syllabus
2023-24

Chapter 5

Problem Solving

Computer Science

Class XI (As per CBSE Board)

Visit : python.mykvs.in for regular updates

Problem Solving

What is problem

Problems are at the center of what many people do at work every day.

A matter or situation regarded as unwelcome / harmful and need to be dealt with overcome is known as problem.

Computer / Microprocessor is the main device nowadays for finding the solution of a problem ,because of it's speed and the way it can deal with data.

Generally problems are solved using some sort of program/software using any computer programming language.

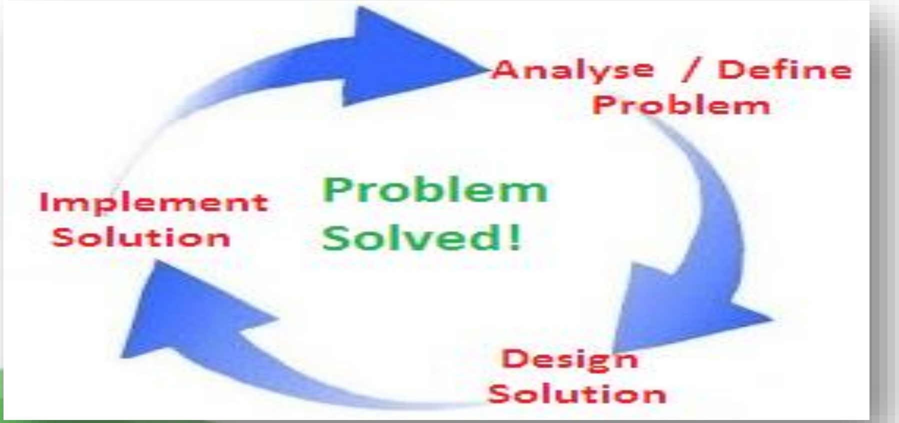
Problem Solving

Problem Solving Cycle

There are 3 basic steps for solving any problem using computer/computer program

1. Analyse / Define problem
2. Design Solution (developing an algorithm, coding, testing and debugging)
3. Implement solution

   Which can be further extended in larger domain.


Analyse / Define Problem
Design Solution
Implement Solution
Problem Solved!

Problem Solving Cycle

1. Analyse / Define problem

In almost every problem solving methodology the first step is defining or identifying the problem. It is the most difficult and the most important of all the steps. It involves diagnosing the situation so that the focus on the real problem and not on its symptoms.

For example, ‘if performance in any department is substandard, we might think the problem is with the individuals submitting work. However, if we look a bit deeper, the real issue might be a lack of training, or an unreasonable workload’.

As above example, same thing may happen with problem solving using computer program, So before starting designing/coding ,problem must be thoroughly defined/ analysed.

Problem Solving

Problem Solving Cycle

1. Analyse / Define problem

The process of understanding the problem and then defining it on the basis of following:

- Data requirement of the given problem
- Type of input variable required
- Type of output variable required is called problem definition

Problem Solving

Problem Solving Cycle

1. Analyse / Define problem

Suppose we are assigned to solve an arithmetic problem and we are not familiar with steps involved in solving that problem. In Such a case,we will not be able to solve the problem. The same applied to writing computer programs. A computer programmer cannot write the instructions to be followed by the computer unless he knows how to solve that problem. The first step in application development is problem definition. In addition, a program souce data, logical and practical procedures needed to solve the problem. This analysis must taken placed before actual preparation of program development, otherwise it will cause much cost and time.

Problem Solving

Problem Solving Cycle

2. Design Solution

Design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

Programming tools, program design tools are the tools used to develop a program. following are some of them

- Algorithm
    - Flowchart
    - Pseudo-code

Problem Solving Cycle

2. Design Solution

Designing an algorithm:

An algorithm is a plan, a logical step-by-step process for solving a problem. Algorithms are normally written as a flowchart or in pseudo-code as it’s next level.

When designing an algorithm there are two main areas to look at:

- the big picture - What is the final goal?
- the individual stages – What hurdles need to be overcome on the way to the goal?

Problem Solving

Problem Solving Cycle

2. Design Solution

An algorithm to Find largest of two numbers

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

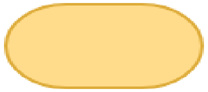
Problem Solving

A **flowchart** is simply a graphical representation of steps. It shows steps in a sequential order, and is widely used in presenting flow of algorithms, workflow or processes. Typically, flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.

Flowchart Symbols

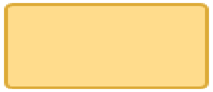
Different flowchart shapes have different conventional meanings. The meanings of some of the more common shapes are as follows:

1. Terminator



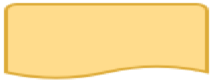
The terminator symbol represents the starting or ending point of the system.

2. Process



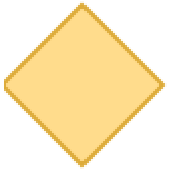
A box indicates some particular operation.

3. Document



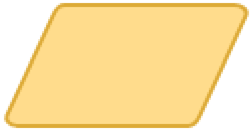
This represents a printout, such as a document or a report.

Problem Solving



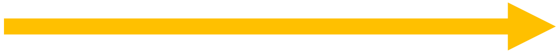
4. Decision

A diamond represents a decision or branching point. Lines coming out from the diamond indicates different possible situations, leading to different sub-processes.



5. Data

It represents information entering or leaving the system. An input might be an order from a customer. An output can be a product to be delivered.



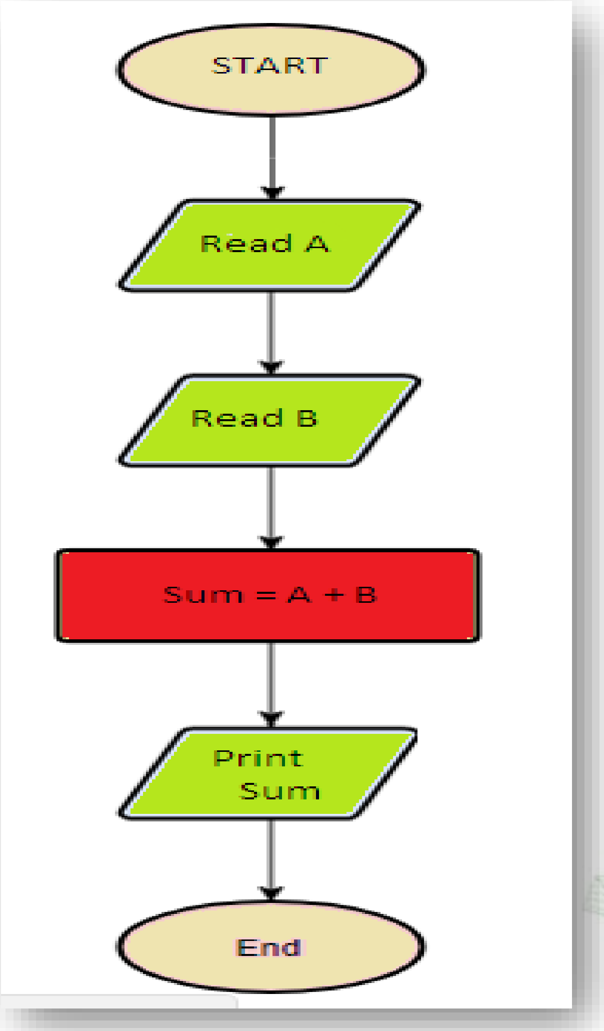
6. Flow

Lines represent flow of the sequence and direction of a process.

Problem Solving

Representation of algorithm using flowchart

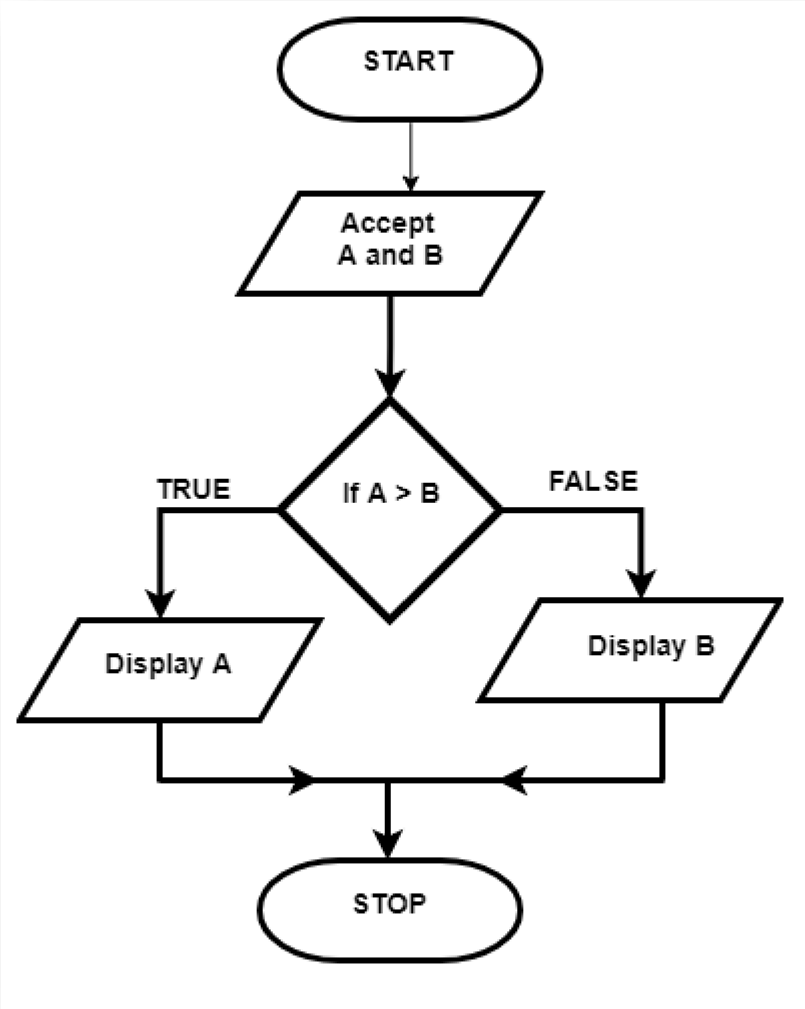
E.g. Flowchart for addition of two numbers


START
Read A
Read B
Sum = A + B
Print Sum
End

Problem Solving

Representation of algorithm using flowchart

Example- Flowchart for finding largest of two numbers


START
Accept
A and B
If A > B
TRUE
FALSE
Display A
Display B
STOP

Problem Solving

Pseudo-code

Pseudocode is not a programming language, it is a simple way of describing a set of instructions that does not have to use specific syntax.

There is no strict set of standard notations for pseudocode, but some of the most widely recognised are:

INPUT/READ – indicates a user will be inputting something

OUTPUT/WRITE – indicates that an output will appear on the screen

WHILE – a loop (iteration that has a condition at the beginning)

FOR – a counting loop (iteration)

REPEAT – UNTIL – a loop (iteration) that has a condition at the end

IF – THEN – ELSE – a decision (selection) in which a choice is made

any instructions that occur inside a selection or iteration are usually indented

Problem Solving

Representation of algorithm using Pseudo-code

Example- Pseudocode to find out largest of two numbers

```
Write "please enter 2 numbers"
Read n1,n2
If(n1 > n2)
 Set max to n1
Else
 Set max to n2
```

Difference between algorithm and pseudo-code

- An algorithm is a well defined sequence of steps that provides a solution for a given problem, while a pseudocode is one of the methods that can be used to represent an algorithm.
- While algorithms can be written in natural language, pseudocode is written in a format that is closely related to high level programming language structures.
- Pseudocode does not use specific programming language syntax and therefore could be understood by programmers who are familiar with different programming language.Transforming an algorithm presented in pseudocode to programming code could be much easier than converting an algorithm written in natural language.

Problem Solving

Coding :

coding is basically implementing logic/algorithm/pseudocode (derived from requirement analysis/problem definition) in one of the preferred programming language(C,C++ Java, Javascript, python etc) as per the protocols/rules/syntactic grammar of the choosen language by following the design decisions.

Problem Solving

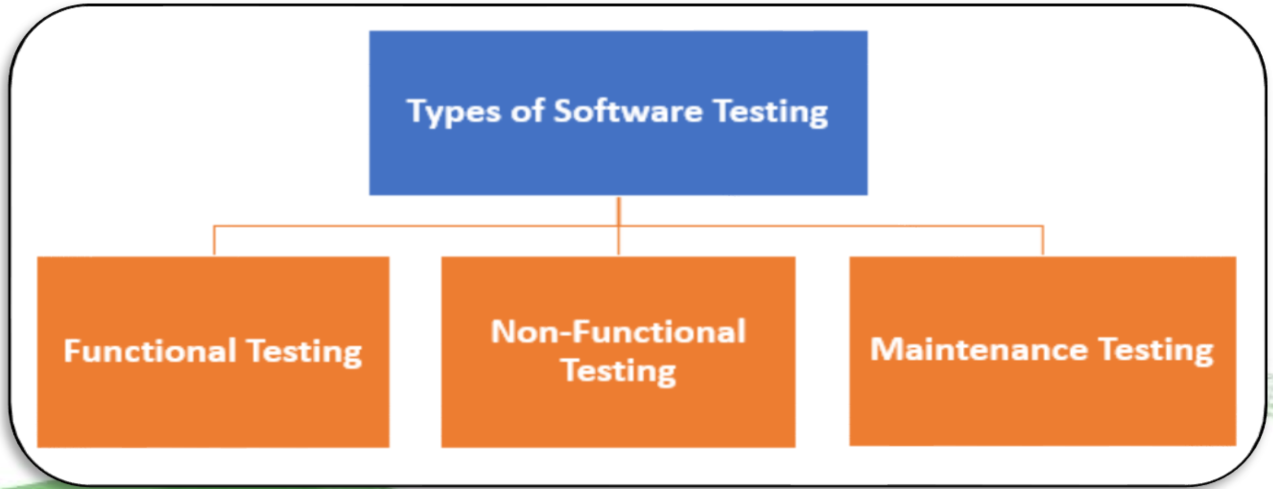
Testing :

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction

Types of Software Testing

- Functional Testing
- Non-Functional Testing or Performance Testing
- Maintenance (Regression and Maintenance)


Types of Software Testing
Functional Testing
Non-Functional Testing
Maintenance Testing

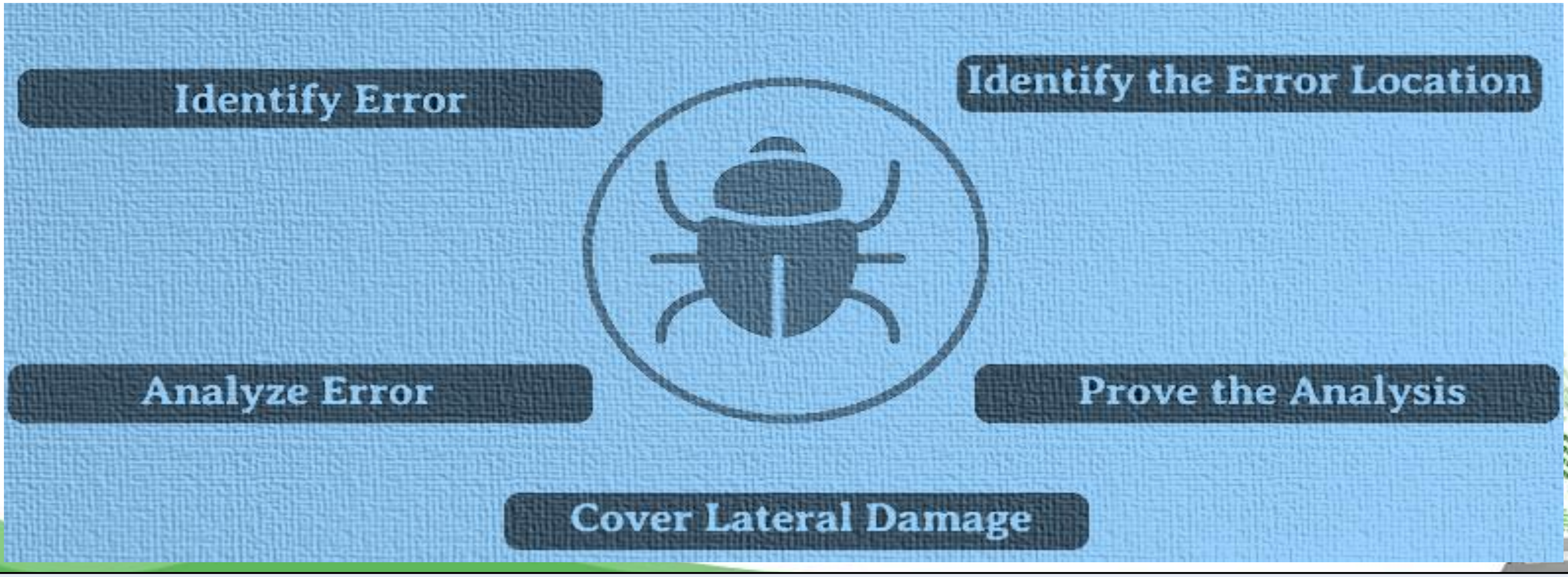
Types of Software Testing

Testing	Types of Testing
Functional Testing	Unit Testing - individual units or components of a software are tested Integration Testing –components are combined and tested as a group Smoke - determines whether the employed build is stable or not UAT (User Acceptance Testing) - actual users test the software as was required Localization- behavior of a software is tested for a specific region,locale ,culture Globalization - ensure that the software application can work in any locale
Non-Functional Testing	Performance - for testing the speed, response time, stability,reliability, scalability Endurance - testing a system with a significant load given over a time Load - software application is tested under a specific expected load Volume - to check the data volume handled by software Scalability - ability to scale up or scale down the number of user requests Usability - how easy and user-friendly a software application is
Maintenance	Regression - testing existing software applications to make sure that a change or addition hasn't broken any existing functionality Maintenance - Testing done during this enhancement, change and migration cycle is known as maintenance testing

Debugging

Software programs goes through testing, updating, troubleshooting, and maintenance during the development process. Usually, software contains errors and bugs, which are removed routinely. Debugging is the process of fixing a bug in the software.

Debugging Steps ->


Identify Error
Identify the Error Location
Analyze Error
Prove the Analysis
Cover Lateral Damage

3. Implementation

Implementation refers to the process of adopting and integrating a software application into a real environment.Implementation of new tools and software into an enterprise can be complex, depending on the size of the organization and the software.

Implementation Methods

Parallel- When the new system is used at the same time as the old system the two systems are said to be running in parallel.

Phased-When small parts of the new system gradually replace small parts of the old system, the implementation method is said to be phased.

Pilot-When a small group of users within an organization uses a new system prior to wider use, the system is said to be piloted.

Direct-When a new system is implemented without any phased or pilot implementation

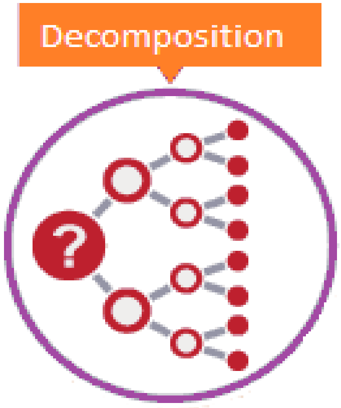
Problem Solving

Decomposition

Decomposition also known as factoring, is breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain.

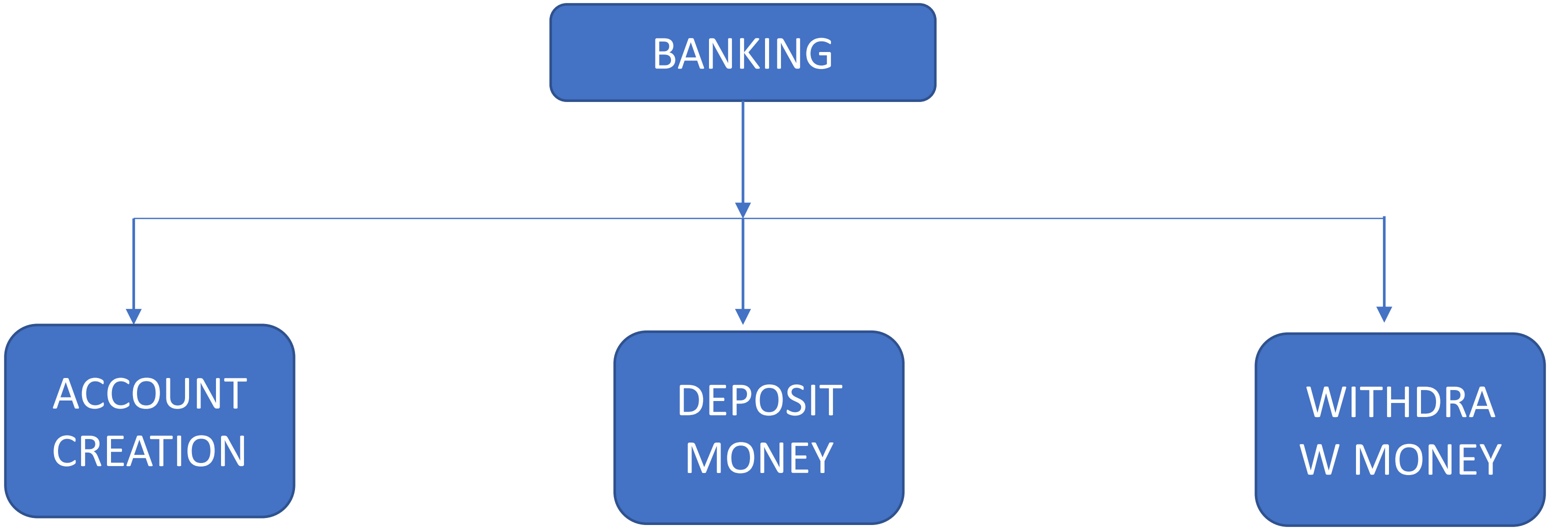
Need for decomposing a problem

It involves breaking down a complex problem or system into smaller parts that are more manageable and easier to understand. The smaller parts can then be examined and solved, or designed individually, as they are simpler to work with.



Example of Decomposition

1. Banking Transaction System



BANKING

ACCOUNT CREATION

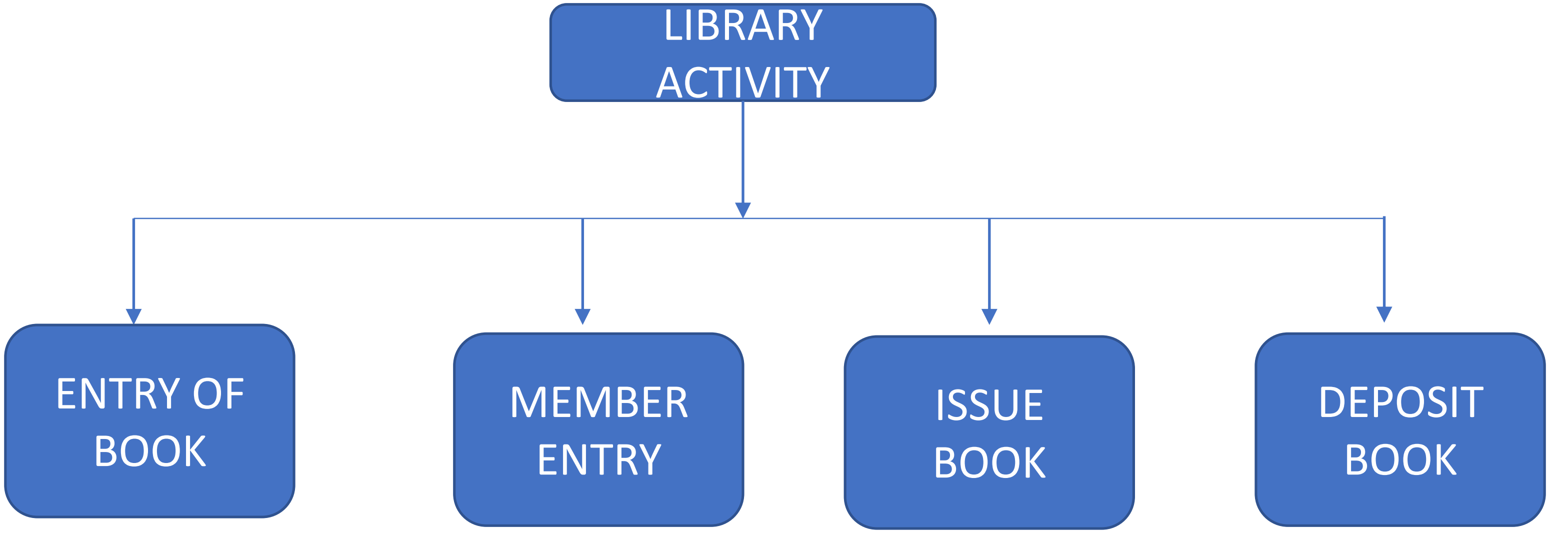
DEPOSIT MONEY

WITHDRAW MONEY

Note : It can be further decomposed to next level as per need

Example of Decomposition

2. LIBRARY MANAGEMENT SYSTEM


LIBRARY ACTIVITY

ENTRY OF BOOK

MEMBER ENTRY

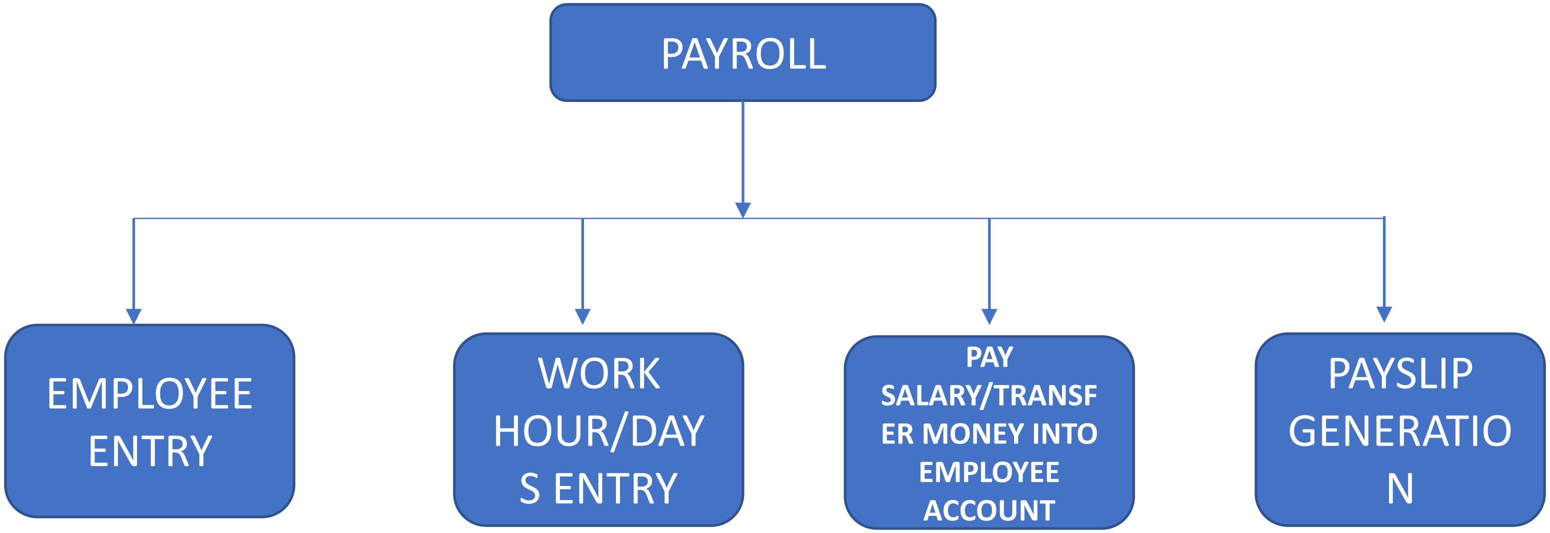
ISSUE BOOK

DEPOSIT BOOK

Note : It can be further decomposed to next level as per need

Example of Decomposition

3. PAYROLL SYSTEM



PAYROLL

EMPLOYEE ENTRY

WORK HOUR/DAYS ENTRY

PAY SALARY/TRANSFER MONEY INTO EMPLOYEE ACCOUNT

PAYSLIP GENERATION

Note : It can be further decomposed to next level as per need