Problem Solving and Programming

CSE1001

Skills Required for a Software Engineer

Technical Skills

- Software Design
- Coding
- Testing

Problem Solving Skills

- logical and analytical thinking

Soft Skills

- Communication
- Team Work

How do you define a “Problem”?

- Problem is a puzzle that requires logical thought and /or mathematics to solve.
- A puzzle could be a set of questions on a scenario which consists of ***description of reality*** **and set of constraints about the scenario.**
- **Example Scenario:** VIT Chennai campus has a library. The librarian issues books only to VIT employees.
- **Description of reality** : There is a library in VIT Chennai campus and there is a librarian and collection of books in the library.

Problem ?

- **Constraint** : Librarian issues books only to VIT employees.

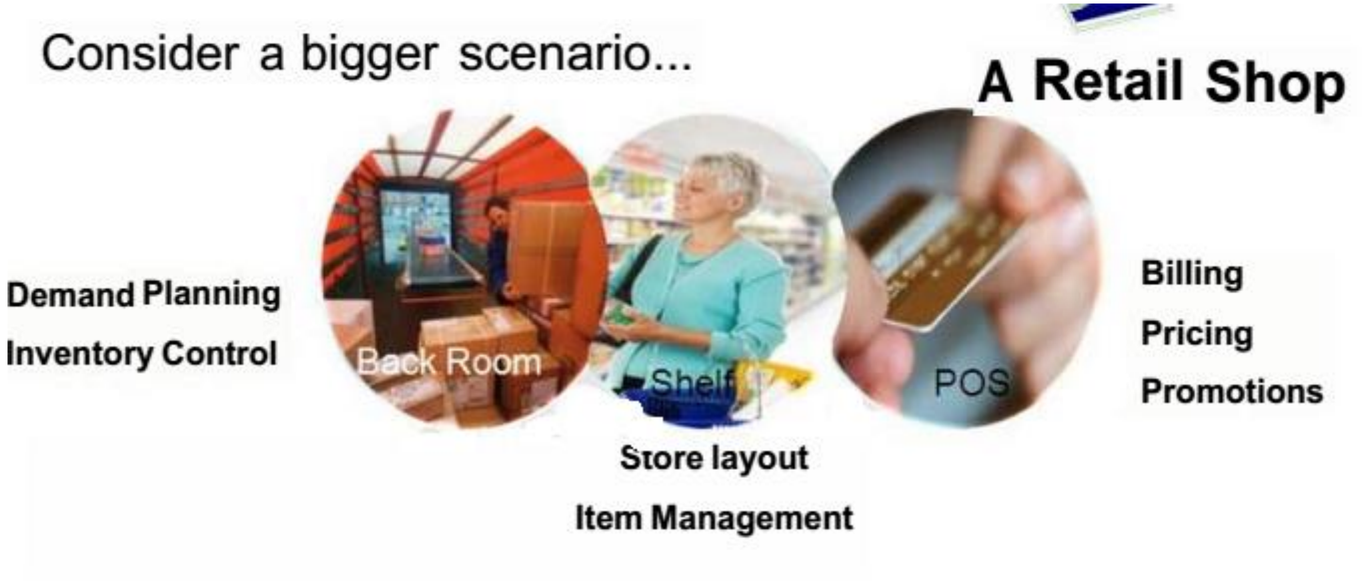
  Questions about this scenario:

  1 How many books are there in the library?

  2 How many books can be issued to an employee?

  3. Does the librarian issue a book to himself? etc

Case study - Discussion



Have you ever observed this scenario?

Yes!!! What are the problems in the scenario?

Types of Problems

- All Problems do not have straightforward solutions.
- Some problems, such as balancing a checkbook or baking a cake, can be solved with a series of actions.
- These solutions are called **algorithmic solutions**.
- There may be more than one solution for a problem
- Identify all possible ways to solve a problem and choose the best one among them(best – depends)

Types of Problems

- The solutions of other problems, such as how to buythe best stock or whether to expand the company, are not, so straight forward.
- These solutions require reasoning built on knowledge and experience, and a process of trial and error.
- Solutions that cannot be reached through a direct set of steps are called **heuristic solution.**

Problem Solving with Computers

- Computers are built to solve problems with algorithmic solutions, which are often difficult or very time consuming when input is large.
- Solving a complicated calculus problem or alphabetizing 10,000 names is an easy task for the computer.
- So the basis for solving any problem through computers is by developing an appropriate and efficient algorithm.

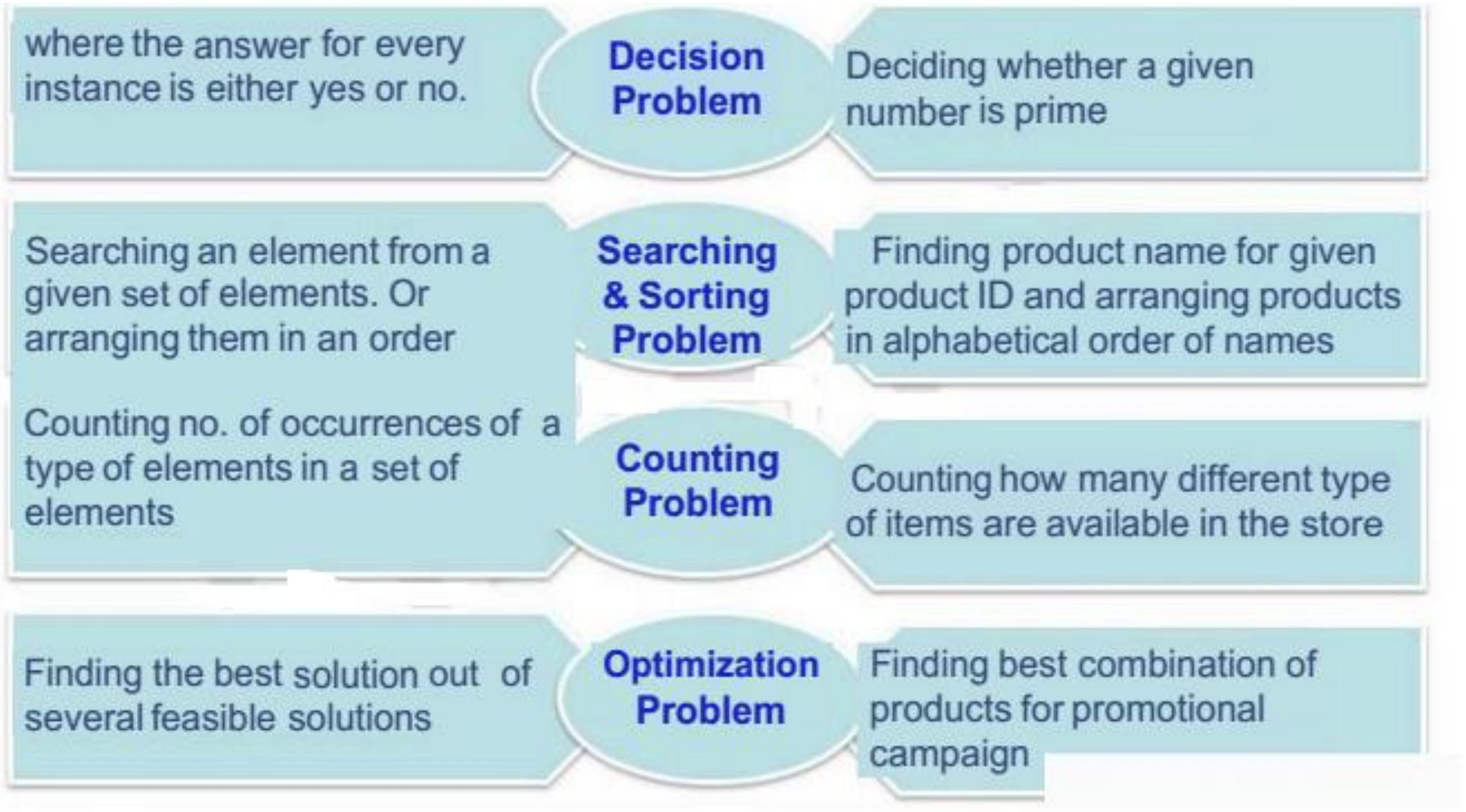
Problem Solving with Computers

- Field of computers that deals with heuristic types of problems is called Artificial Intelligence (AI).
- Artificial intelligence enables a computer to do things like human by building its own knowledge bank.
- As a result, the computer's problem-solving abilities are similar to those of a human being.
- Artificial intelligence is an expanding computer field, especially with the increased use of Robotics.

Computational Problems

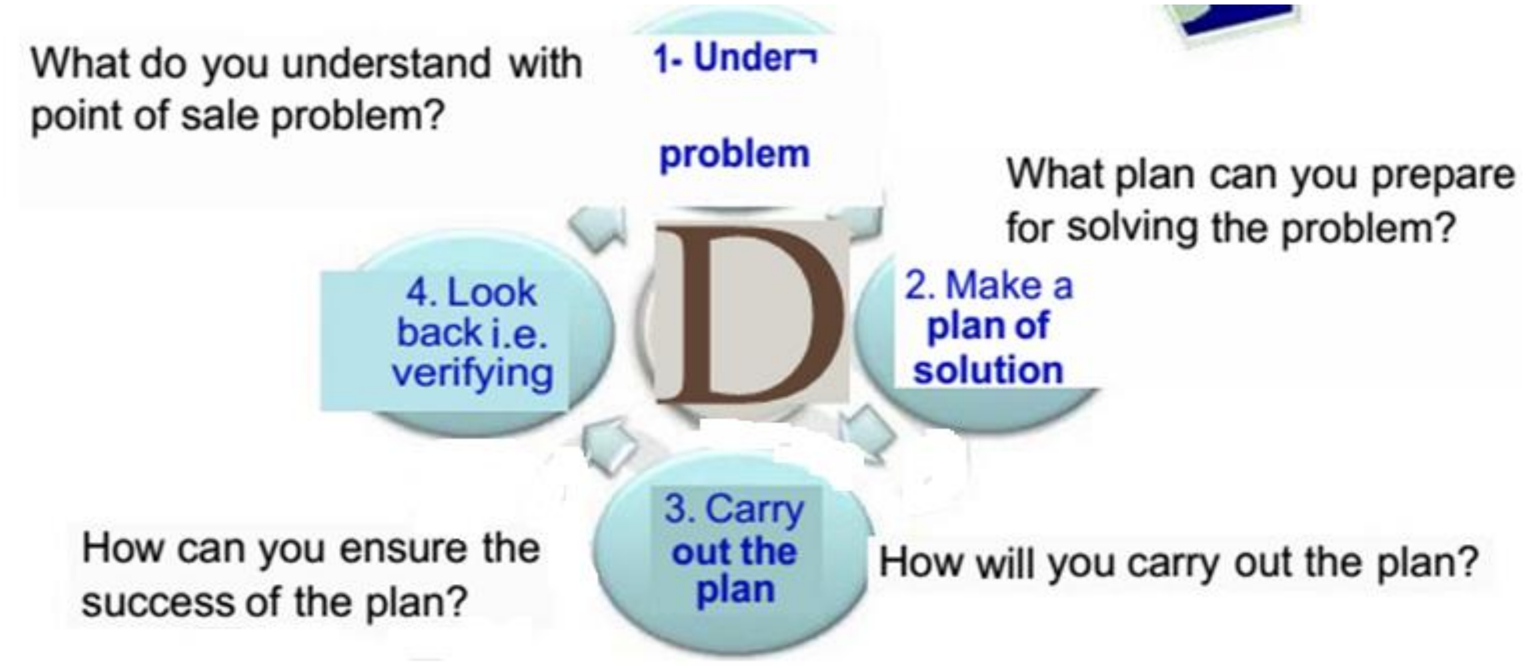
- Computation is the process of evolution from one **state to** another in accordance with some specified **rules.**

Types of Computational Problems



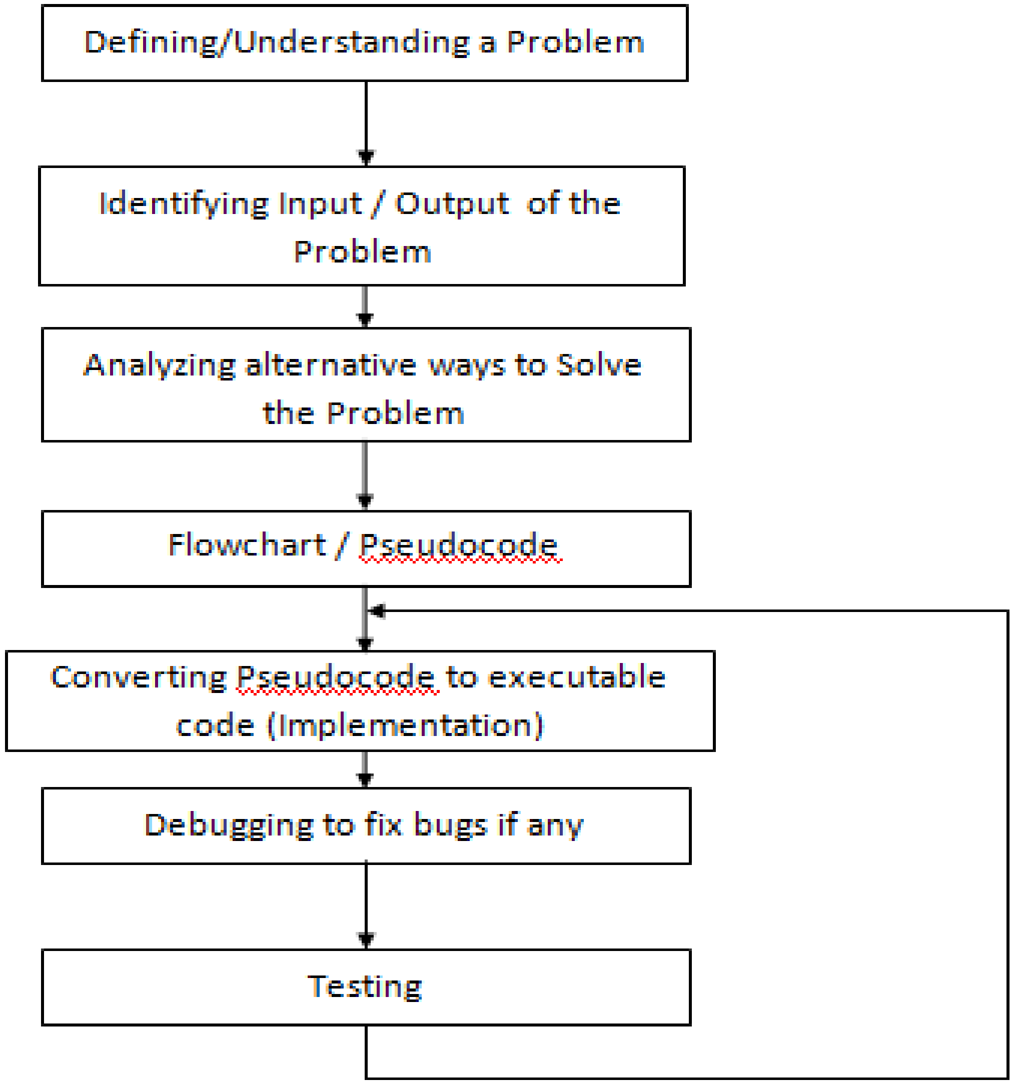
where the answer for every instance is either yes or no.	**Decision Problem**	Deciding whether a given number is prime
Searching an element from a given set of elements. Or arranging them in an order	**Searching & Sorting Problem**	Finding product name for given product ID and arranging products in alphabetical order of names
Counting no. of occurrences of a type of elements in a set of elements	**Counting Problem**	Counting how many different type of items are available in the store
Finding the best solution out of several feasible solutions	**Optimization Problem**	Finding best combination of products for promotional campaign

Problem Solving Life Cycle


What do you understand with point of sale problem?
1- Under
problem
What plan can you prepare for solving the problem?
D
2. Make a plan of solution
4. Look back i.e. verifying
3. Carry out the plan
How can you ensure the success of the plan?
How will you carry out the plan?

For any problem solving strategy logic is prerequisite.

Problem Solving Life Cycle


Defining/Understanding a Problem
Identifying Input / Output of the Problem
Analyzing alternative ways to Solve the Problem
Flowchart / Pseudocode
Converting Pseudocode to executable code (Implementation)
Debugging to fix bugs if any
Testing

Logic – Basis for solving any problem

- **Definition : A method of human thought that involves thinking in a linear, step by step manner about how a problem can be solved**
- Logic is a language for reasoning.
- It is a collection of rules we use when doing reasoning.
- Eg: John's mum has four children.
- The first child is called April.
- The second May.
- The third June.
- What is the name of the fourth child?

What Problem Can Be Solved By Computer

- Solving problem by computer undergo two phases:
  - Phase 1:
    - Organizing the problem or pre-programming phase.
  - Phase 2:
    - Programming phase.

PRE-PROGRAMMING PHASE

- **Analyzing The Problem**
  - Understand and analyze the problem to determine whether it can be solved by a computer.
  - Analyze the requirements of the problem.
  - Identify the following:
    - Data requirement.
    - Processing requirement or procedures that will be needed to solve the problem.
    - The output.

PRE-PROGRAMMING PHASE

- All these requirements can be presented in a Problem Analysis Chart (PAC)

Data	Processing	Output	Solution Alternatives
given in the problem or provided by the user	List of processing required or procedures.	Output requirement.	List of ideas for the solution of the problem.

PRE-PROGRAMMING PHASE

- **Payroll Problem**
  - Calculate the salary of an employee who works by hourly basis. The formula to be used is

Salary = Hour works * Pay rate

Data	Processing	Output	Solution Alternatives
Hours work, Pay rate	Salary = Hours work * payrate	Salary	1. Define the hours worked and pay rate as constants. ∗2. Define the hours worked and pay rate as input values.

Miles to Km

Write a Problem Analysis Chart (PAC) to convert the distance in miles to kilometers where 1.609 kilometers per mile.

Data	Processing	Output	Solution Alternatives
Distance in miles	Kilometers = 1.609 x miles	Distance in kilometers	1. Define the miles as constants. ∗2. Define the miles as input values.

Importance of Logic in problem solving

Determine whether a given number is prime or not?

Data	Processing	Output	Solution Alternatives
Number, N	Check if there is a factor for N	Print Prime or Not Prime	1. Divide N by numbers from 2 to N and if for all the division operations, the reminder is non zero, the number is prime otherwise it is not prime 2. Same as 1 but divide the N from 2 to N/2 3. Same as Logic 1 but divide N from 2 to square root of N

Importance of Logic in problem solving

In a fun game, MXM grid is given with full of coins. The player has to give a number 'N' of his choice. If N is lesser than M then he is out of game and doesn't gain any points. Otherwise he has to place all coins in the MXM grid in the NXN grid and he gains points equal to the number of free cells in the N X N grid.

Data	Processing	Output	Solution Alternatives
Numbers M and N	If N is less than M Points = 0 Otherwise Compute Points as N^2 - M^2	Number of points gained	1. Compute N^2 - M^2 as NXN – MXM 2. Compute (N + M) X (N – M) (Number of multiplication is reduced)

Problem 2

- Write a Problem Analysis Chart (PAC) to find an area of a circle where area = pi * radius * radius

Data	Processing	Output
radius	area = 3.14 x radius x radius	area

Problem 3

- Write a Problem Analysis Chart (PAC) to compute and display the temperature inside the earth in Celsius and Fahrenheit. The relevant formulas are

  Celsius = 10 x (depth) + 20

  Fahrenheit = 1.8 x (Celsius) + 32

Data	Processing	Output

depth	celsius = 10 x (depth) + 20
	fahrenheit = 1.8 x (celsius) + 32
		Display celsius,
		Display fahrenheit

Problem 4

Given the distance of a trip in miles, miles per gallon by the car that is used in the trip and the current price of one gallon of fuel (gas). Write a program to determine the fuel required for the trip and the cost spent on the fuel.

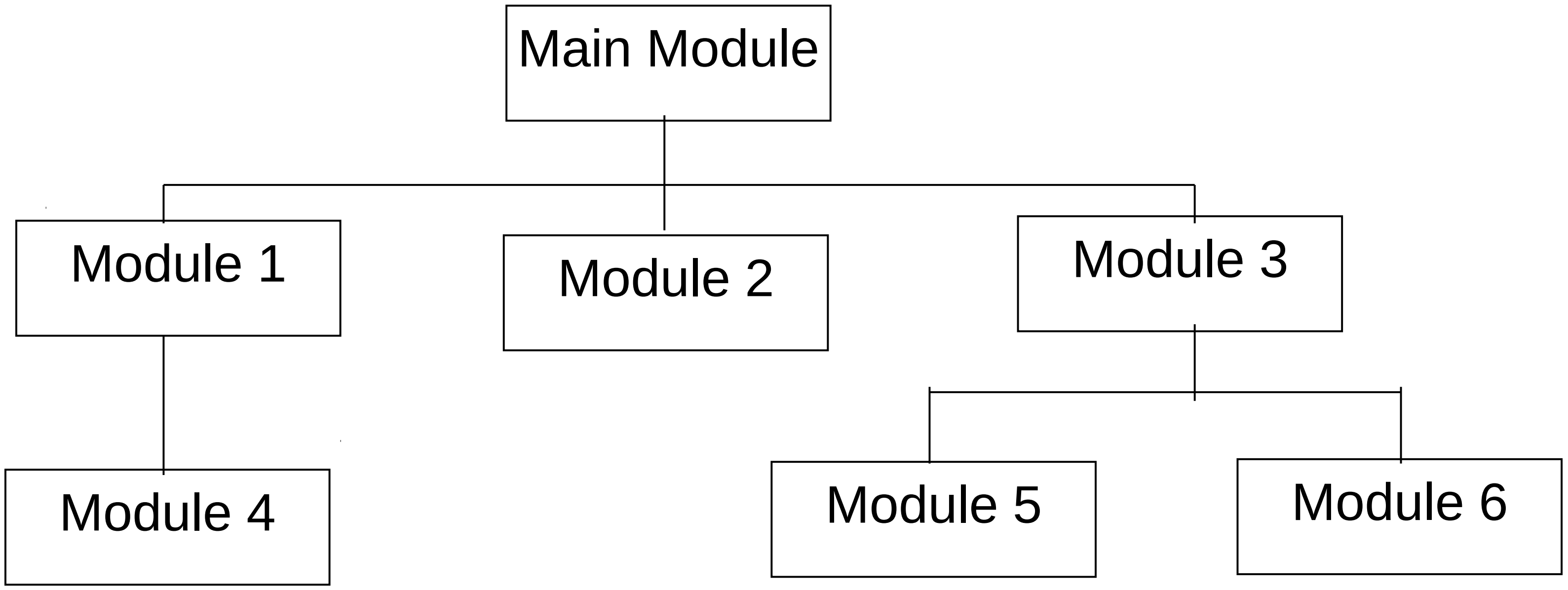
Input	Processing	Output
Distance in miles, miles per gallon, cost per gallon	gas needed = distance / miles per gallon. estimated cost = cost per gallon x gas needed	Display gas needed Display estimated cost

HIPO Chart

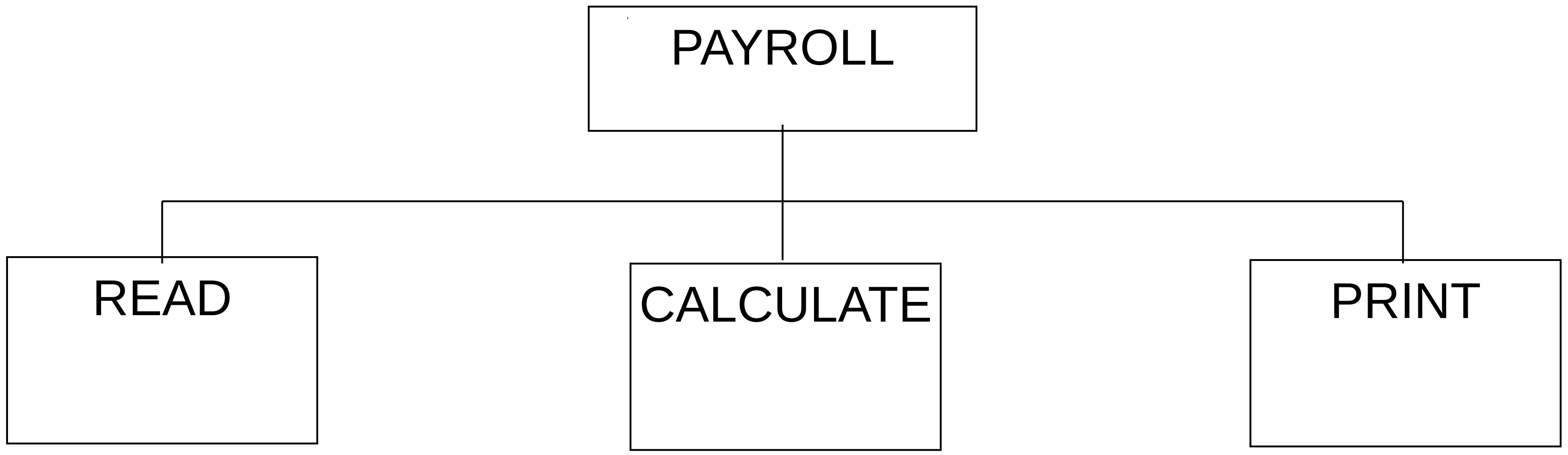
- **Developing the Hierarchy Input Process Output (HIPO) or Interactivity Chart**
  - When problem is normally big and complex.
  - Processing can be divided into subtasks called modules.
  - Each module accomplishes a specific task.
  - These modules are connected to each other to show the interaction of processing between the modules.

HIPO Chart

- Programming which use this approach (problem is divided into subtasks) is called *Structured Programming.*



HIPO Chart for Payroll Problem



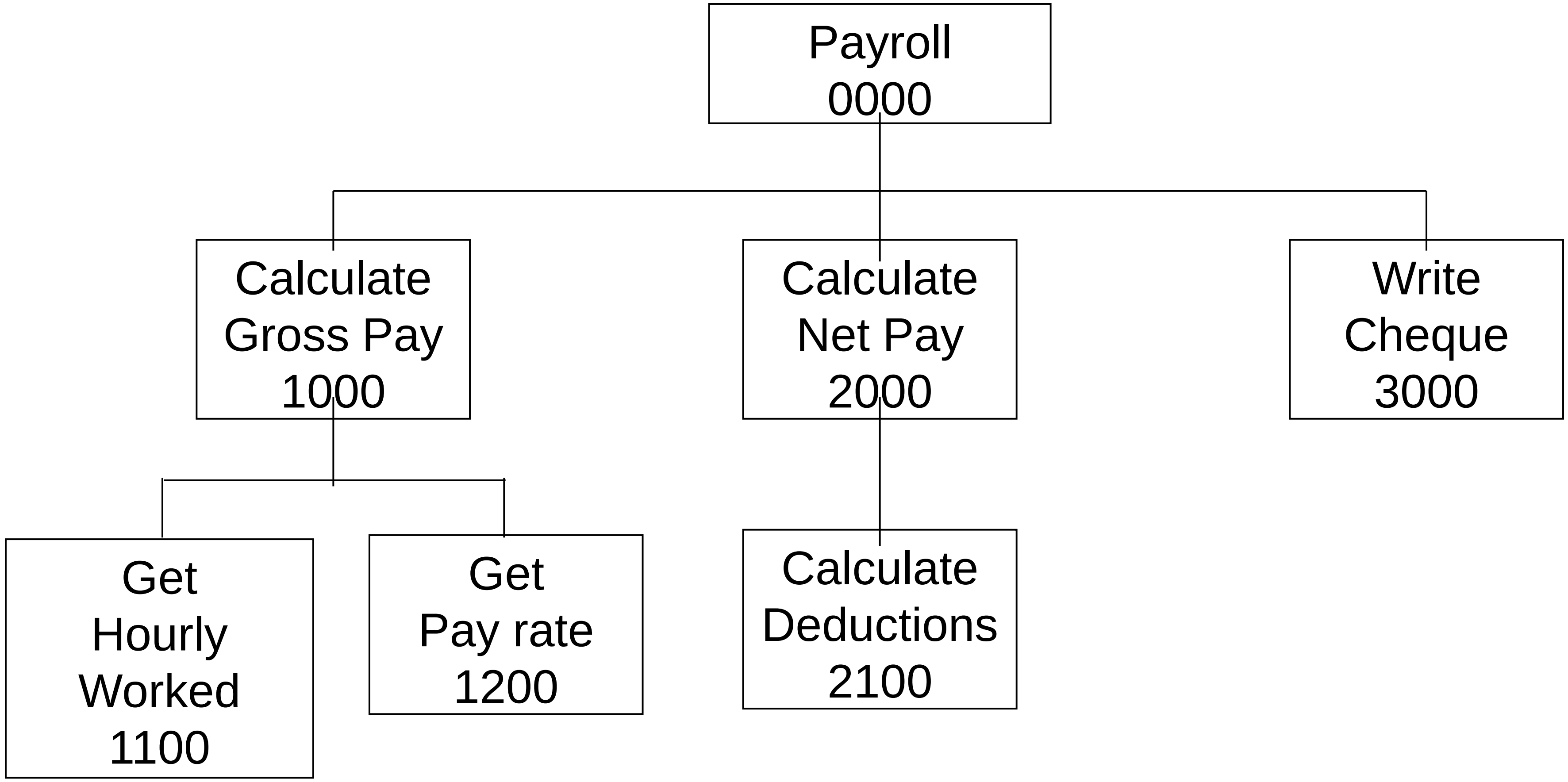
Extended Payroll Problem

- You are required to write a program to calculate both the gross pay and the net pay of every employee of your company.
- Use the following formulae for pay calculation:
  - Gross pay = number of hours worked * pay rate
  - Net pay = gross pay – deductions
- The program should also print the cheque that tells the total net pay.

PAC for Extended Payroll Problem

Input	Processing	Output
Number of hours worked, pay rate, deductions	Gross pay = number of hours * pay rate Net pay = Gross pay – deductions	Net pay and write net pay in cheque

HIPO Chart

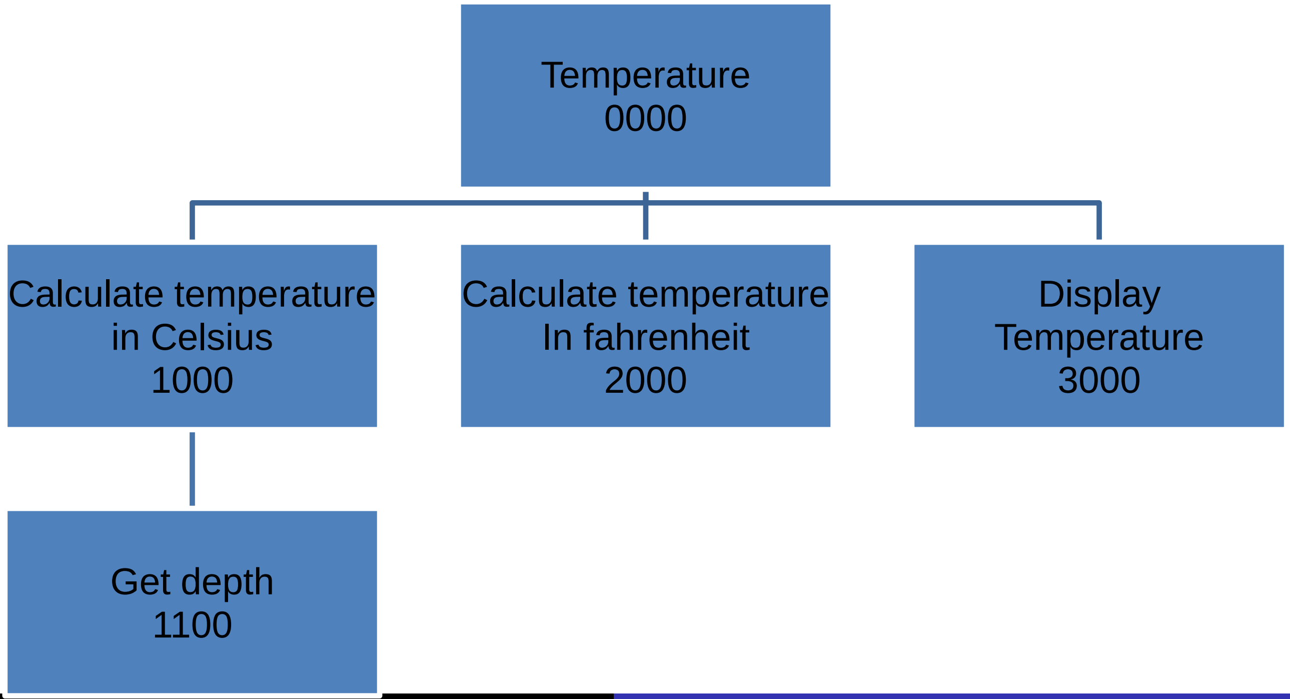

Payroll
0000
Calculate Gross Pay 1000
Calculate Net Pay 2000
Write Cheque 3000
Get Hourly Worked 1100
Get Pay rate 1200
Calculate Deductions 2100

Temperature of Earth

- Write a Hierarchy Input Process Output (HIPO) to compute and display the temperature inside the earth in Celsius and Fahrenheit. The relevant formulas are:

  Celsius = 10 x (depth) + 20

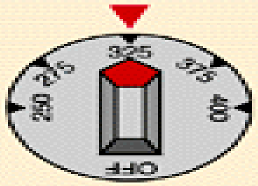
  Fahrenheit = 1.8 x (Celsius) + 32


Temperature
0000
Calculate temperature in Celsius
1000
Calculate temperature In fahrenheit
2000
Display Temperature
3000
Get depth
1100

Algorithm

- Step by step procedure to solve a problem
- 'In Computer Science following notations are used to represent algorithm
- Flowchart: This is a graphical representation of computation.
- Pseudo code: They usually look like English statements but have additional qualities.

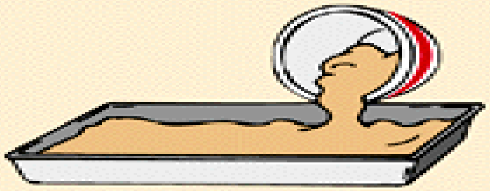
Heat oven to 325°F



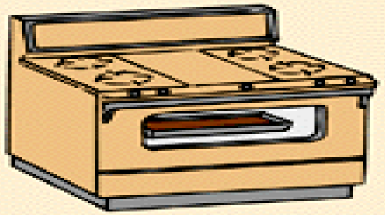
Gather the ingredients

Mix ingredients thoroughly
in a bowl

Pour the mixture into a
baking pan



Bake in the oven
50 minutes

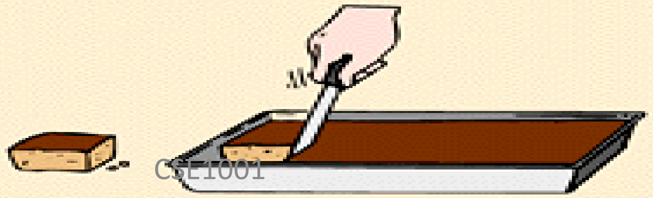


Repeat

Bake 5 minutes more

Until cake top springs back when touched in the center

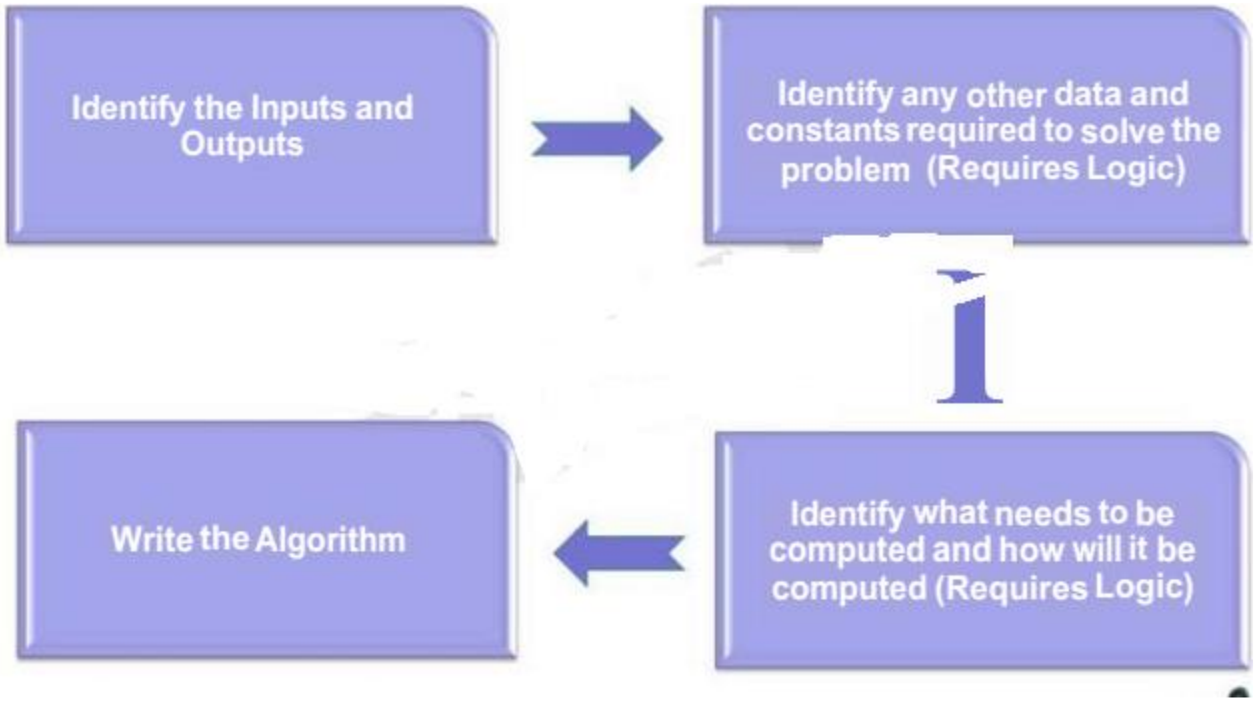
Cool on a rack before cutting



Algorithm

- Algorithms are not specific to any programming language
- An algorithm can be implemented in any programming language
- Use of Algorithms
  - Facilitates easy development of programs
  - Iterative refinement
  - Easy to convert it to a program
  - Review is easier

Steps to Develop an Algorithm


Identify the Inputs and Outputs
Identify any other data and constants required to solve the problem (Requires Logic)
Identify what needs to be computed and how will it be computed (Requires Logic)
Write the Algorithm

Algorithm for Real life Problem

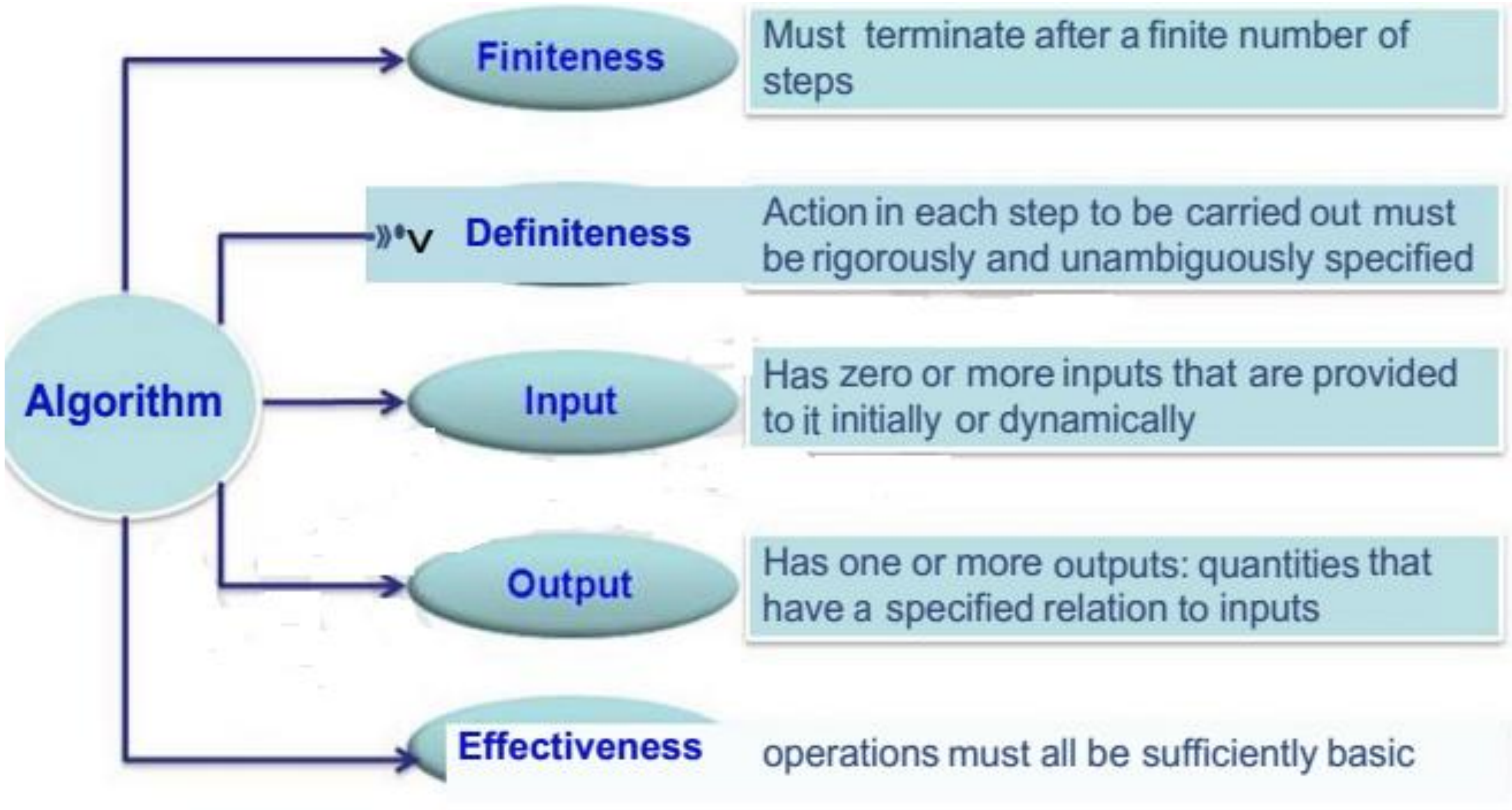
PROBLEM: Heat up a can of soup

ALGORITHM:

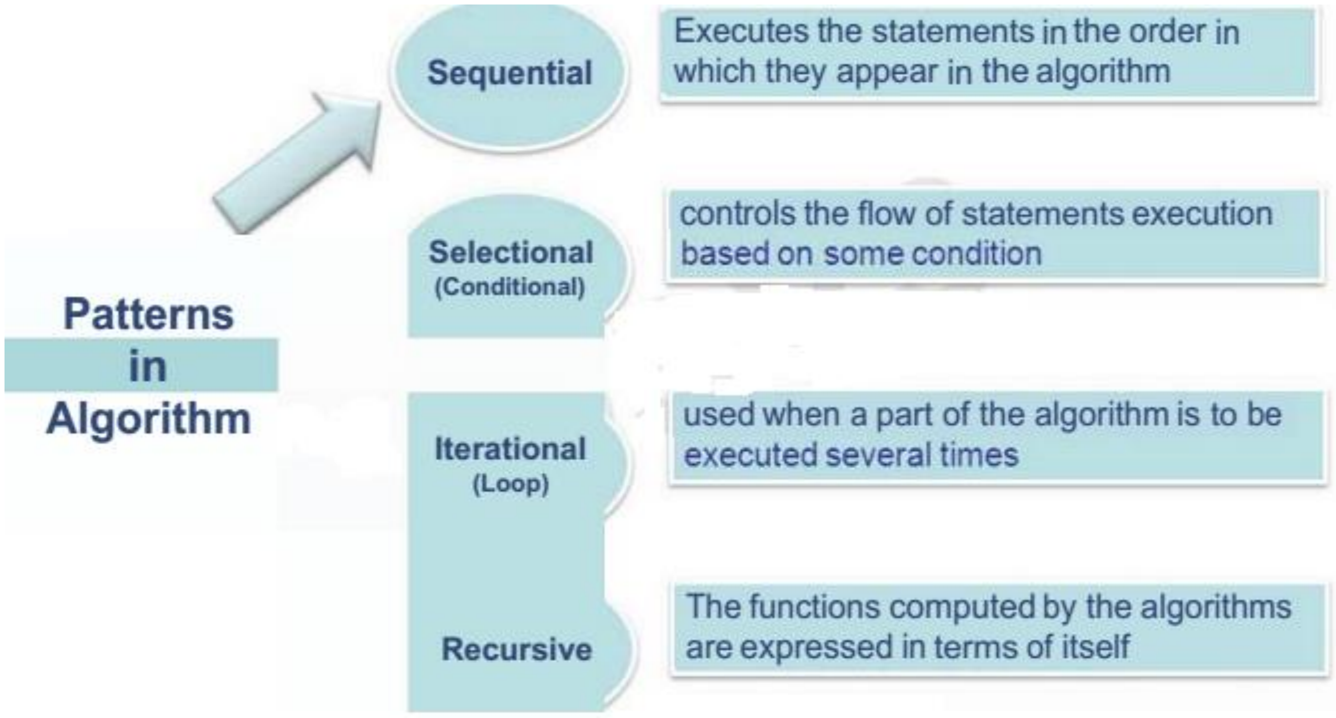
- 1 - open can using can opener
- 2 - pour contents of can into saucepan
- 3 - place saucepan on ring of cooker
- 4 - turn on correct cooker ring
- 5 - stir soup until warm

may seem a bit of a silly example but it does show us that the order of the events is important since we cannot pour the contents of the can into the saucepan before we open the can.

Properties of an Algorithm


Algorithm
Finiteness
Must terminate after a finite number of steps
Definiteness
Action in each step to be carried out must be rigorously and unambiguously specified
Input
Has zero or more inputs that are provided to it initially or dynamically
Output
Has one or more outputs: quantities that have a specified relation to inputs
Effectiveness
operations must all be sufficiently basic

Different patterns in Algorithm


Patterns in Algorithm
Sequential
Executes the statements in the order in which they appear in the algorithm
Selectional (Conditional)
controls the flow of statements execution based on some condition
Iterational (Loop)
used when a part of the algorithm is to be executed several times
Recursive
The functions computed by the algorithms are expressed in terms of itself

Sequential Algorithms

Algorithm for adding of two numbers

Step 1: Read two numbers A and B.

Step 2: Let C = A + B.

Step 3: Display C.

Area of a Circle

Step 1 : Read the RADIUS of a circle

Step 2 : Find the square of RADIUS and store it in SQUARE

Step 3 : Multiply SQUARE with 3.14 and store it in AREA

Step 4: Print AREA

Average Marks

- **Find the average marks of a student in 3 subjects:**

```
Step 1 : Read Marks1, Marks2 and  Marks3
Step 2 : Sum = Marks1 + Marks2 + Marks3
Step 3 : Average = Sum / 3
Step 4 : Display Average
```

Selection Algorithms

Algorithm for Conditional Problems

PROBLEM: To decide if a fire alarm should be sounded

ALGORITHM:

1 IF fire is detected	condition
2 THEN sound fire alarm	action

Another example is:-

PROBLEM: To decide whether or not to go to school

ALGORITHM:

```
1 IF it is a weekday AND it is not a holiday
2 THEN go to school
3 ELSE stay at home
```

Pass/ Fail and Average

- Write an algorithm to find the average marks of a student. Also check whether the student has passed or failed. For a student to be declared pass, average marks should not be less than 65.

  Step 1 : Read Marks1, Marks2, Marks3

  Step 2 : Total = Marks1 + Marks2 + Marks3

  Step 3 : Avg = Total / 3

  Step 4 : Set Output = “Student Passed”

  Step 5 : if Avg < 65 then Set Output “Student Failed"

  Step 6 : Display Output

Leap Year or Not

Step 1 : Read YEAR

Step 2 :

```
 IF
({YEAR%4=0 AND YEAR%100!=0)OR (YEAR%400=0))
Display "Year is a leap year"
ELSE
Display "Year is not a leap year"
ENDIF
```

Algorithm for Iterative Problems

This type of loop keeps on carrying out a command or commands UNTIL a given condition is satisfied, the condition is given with the UNTIL command, for example:-

PROBLEM: To wash a car

ALGORITHM:

```
1 REPEAT
2 wash with warm soapy water
3 UNTIL the whole car is clean
```

Iterative Algorithms – Repetitive Structures

- Find the average marks scored by ‘N’ number of students

```
Step 1 : Read the Number Of Students
Step 2 : Let Counter = 1
Step 3 : Read Marks1, Marks2, Marks3
Step 4 : Total = Marks1 + Marks2 + Marks3
Step 5 : Average = Total / 3
Step 6 : Set Output = " Student Passed "
Step 7 : If (Average < 65) then Set Output = " Student Failed"
Step 8 : Display Output
Step 9 : Set Counter = Counter + 1
Step 10 : If (Counter <= NumberOfStudents ) then goto step 3
```

Bigger Problems

- If you are asked to find a solution to a major problem, it can sometimes be very difficult to deal with the complete problem all at the same time.
- For example building a car is a major problem and no-one knows how to make every single part of a car.
- A number of different people are involved in building a car, each responsible for their own bit of the car's manufacture.
- The problem of making the car is thus broken down into smaller and manageable tasks.
- Each task can then be further broken down until we are left with a number of step-by-step sets of instructions in a limited number of steps.
- The instructions for each step are exact and precise.

Top Down Design

- Top Down Design uses the same method to break a programming problem into small manageable steps.
- First of all we break the problem down into smaller steps and then produce a Top Down Design for each step.
- In this way sub-problems are produced which can be refined into manageable steps.

Top Down Design for Real Life Problem

PROBLEM: To repair a puncture on a bike wheel.

ALGORITHM:

1. remove the tyre
2. repair the puncture
3. replace the tyre

Step 1: Refinement:

1. Remove the tyre

    1.1 turn bike upside down

    1.2 lever off one side of the tyre

    1.3 remove the tube from inside the tyre

Step 2: Refinement:

2. Repair the puncture Refinement:

    2.1 find the position of the hole in the tube

    2.2 clean the area around the hole

    2.3 apply glue and patch that place

Step 3: Refinement:

3. Replace the tyre Refinement:

   3.1 push tube back inside tyre

   3.2 replace tyre back onto wheel

   3.3 blow up tyre

   3.4 turn bike correct way up

Still more Refinement:

Sometimes refinements may be required to some of the sub-problems, for example if we cannot find the hole in the tube, the following refinement can be made to **2.1:-**

Still more Refinement:

Step 2.1: Refinement

```
2.1  Find the position of the hole in the tube
    2.1.1   WHILE hole cannot be found
    2.1.2   Dip tube in water
    2.1.3   END WHILE
```

More Problems to Discuss

Problem 1

- Write a program to subtract a vector from another. You are given a program to add two vectors as sample

Problem 1

- We can solve by analogy method

Problem 2

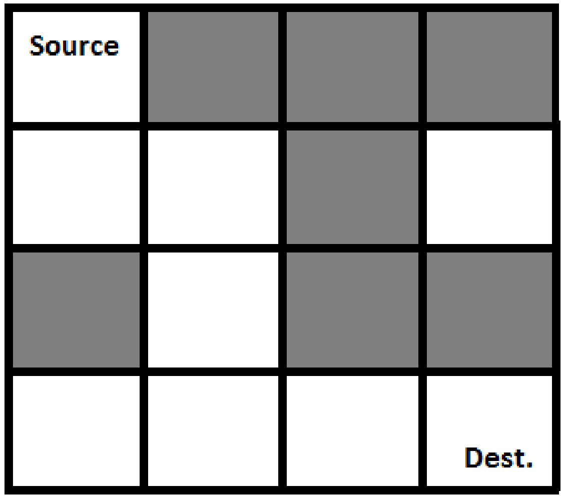
- Persons X and Y can sort a list of numbers. But they have a limitation that they can only sort 100 numbers at a time. Person Z, can combine two sorted list (i.e.) Given two sorted list of numbers L1 and L2, Z can form a bigger list L3 with elements in L1 and L2 in sorted order. In this scenario how will you sort a list of 200 elements

Problem 2

- Divide and conquer

Problem 3

- A Maze is given as N*N binary matrix of blocks where source block is the upper left most block i.e., maze[0][0] and destination block is lower rightmost block i.e., maze[N-1][N-1]. A rat starts from source and has to reach destination. The rat can move only in two directions: forward and down.



Problem 3

- Backtracking
- Down, forward
- Will lead to dead end and rat has to backtrack

Problem 4

- Given a set ‘s’ and a number ‘n’, determine all subsets of set of ‘s’ such that the sum of the elements in the subset is equal to n.

Problem 4

- One of the option is to generate all subset, sum the elements in it and check if it is equal to ‘n’
- Exhaustive search

Problem Solving Strategies

Some of the Problem Solving Strategies are

- Problem Solving by Analogy
- Problem solving by Brainstorming
- Problem solving by Abstraction
- Problem solving by Divide And Conquer
- Problem solving by Exhaustive Search
- Problem solving by Trial And Error
- Problem solving by Backtracking
- Problem solving by Proof
- Problem solving by Reduction
- Problem solving by Hypothesis Testing

Problem solving by Analogy

- An analogy is an inference or an argument from one particular to another particular, where at least one of the premises or the conclusion is general.
- Very often problems can be solved by looking at similar problems.
- **Example**: Consider the problem of adding two matrices A and B of same dimension as 1 x 3.

  i.e Let A =a_{11} a_{12} a_{13} and B =b_{11} b_{12} b_{13}

  Then A + B =$a_{11} + b_{11}$, $a_{12} + b_{12}$, $a_{13} + b_{13}$ we have developed a program for adding the two matrices A and B.
- The program can be easily modified for matrix subtraction.

  We need to do is to replace the + sign by the - sign

  So A - B =$a_{11} - b_{11}$, $a_{12} - b_{12}$, $a_{13} - b_{13}$.

  In problem solving by analogy, we use a solution that solves an analogous problem.

Problem solving by Brainstorming

- It is a group or individual creativity technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its (group) member(s).
- The term was popularized by Alex Faickney Osborn in the book Applied Imagination(1953).
- He claimed that brainstorming was more effective than individuals working alone in generating ideas.
- Example Consider a scenario in a company, when it has got an additional project.

  To manage the extra work, a discussion is made among 3 employees of the company and they give different suggestions.
  - Suggestion - 1: Outsourcing the work.
  - Suggestion - 2: Hire a new employee.
  - Suggestion - 3: Share extra workload among employees.

Problem Solving by Abstraction

- It is used make models that can be used and re-used without having to re-write all the program code for each new application on every different type of computer.
- Suppose we have a quadratic equation $ax^2+bx+c = 0$.
- The roots of the above quadratic Equation are

  $-b \pm \text{sqrt}(b^2-4ac)/2a$.

- Suppose we have been asked to solve the equation $ay^4+by^2+c=0$. substituting y^2 by x $\Rightarrow$ new quadratic equation is $ax^2+bx+2 = 0$. We know how to solve this quadratic equation so we can also solve the equation $ay^4+by^2+c = 0$.
- Abstraction helps to simplify the original problem into problems which are simpler and for which we know the solution.

Problem Solving by Divide and Conquer

- A divide and conquer approach works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly.
- The solutions to the sub-problems are then combined to give a solution to the original problem.

  Example: Summation/Sorting of N – number in given Set.
- After Getting result of each sub problem, it add the solutions of each sub problem two at a time.
- These Solution of sub problem will produce the solution to Main or Original Problem.

Problem Solving by Exhaustive Search

- Suppose we have a Boolean Expression.

  $F(a,b,c) = ab'c' + abc' + ac'$

- We need to Find all possible set of tuples which should yields true for all possibilities to the above Boolean expression.
- So the Possibilities are 2^3. Because, 3 variable are there in Boolean expression.
- If we have n-variables in Boolean expression → 2^n possibilities
- Conclusion: The number of searches grows exponentially based on input variables of Boolean expression.
- This is one of the example of Exhaustive Search.
- The search space grows exponentially based on the input factor.
- The another name is brute-force search.

Problem Solving by Trail and Error

- It is a fundamental method of solving problems.
- It is characterised by repeated, varied attempts which are continued until success.
- It is also a heuristic method of problem solving, repair, tuning, or obtaining knowledge.
- This method is called as Generate and Test in Computer Science Domain.
- Suppose we have to find the factor for a polynomial equation: x^2-5x+6.
- The original binomials must have looked like this: $(\boldsymbol{x}+\boldsymbol{m})(\boldsymbol{x}+\boldsymbol{n})$, where $\boldsymbol{m}$ and $\boldsymbol{n}$ are integers.
- We need to identify the values of $\boldsymbol{m}$ and $\boldsymbol{n}$.
- But the Constant Term of original polynomial is 6 $\rightarrow$ $\boldsymbol{m}*\boldsymbol{n} = 6$. Which integers are used to above equation.
- The Only choices are 2 and 3.
- Trial and error can be used for solving puzzles such as Sudoku.

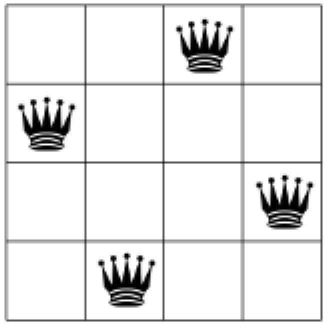
Problem Solving by Backtracking

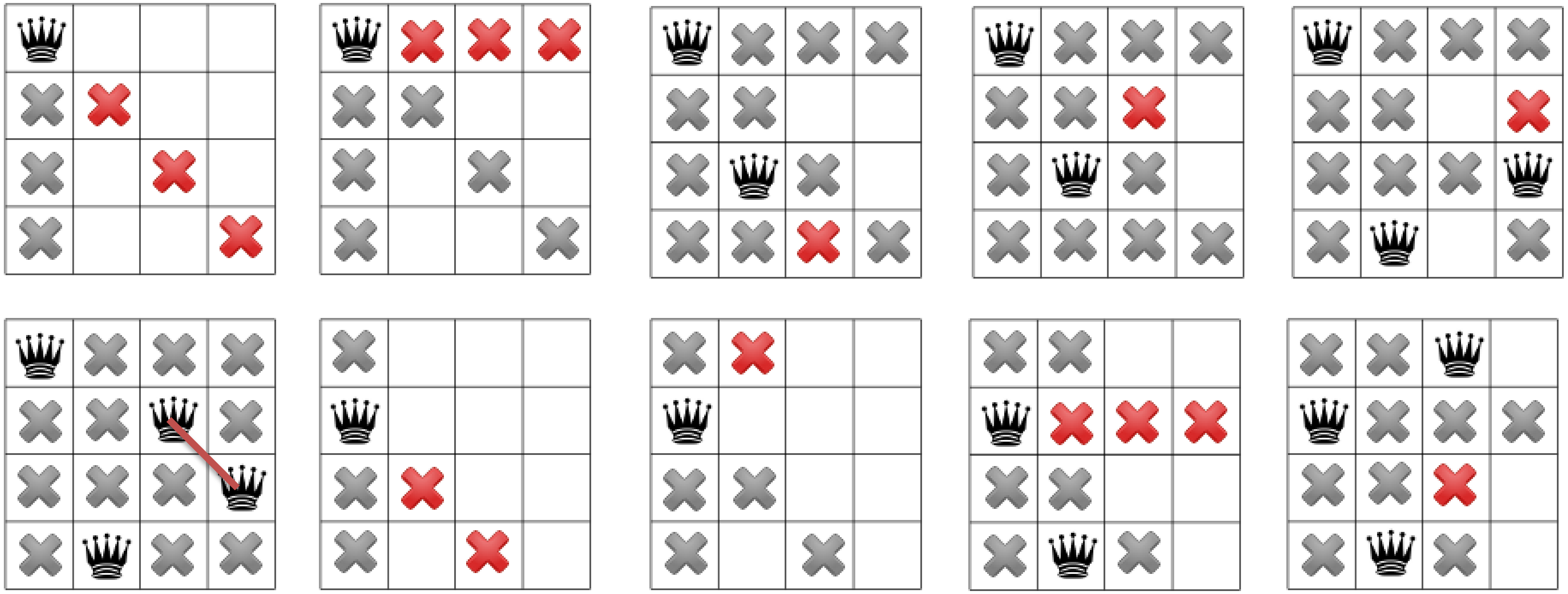
- Suppose we have to arrange 8 queens in a chess board such that no two queens attack each other. This puzzle is called the 8 queens puzzle.
- We can solve this puzzle by considering partial candidates.
- The partial candidates are arrangements of k queens in the first ***k*** rows of the board, all in different rows and columns.
- Any partial solution that contains two mutually attacking queens can be abandoned, since it cannot possibly be completed to a valid solution.
- Backtracking is a problem solving strategy that can be applied only for problems which admit the concept of a ”partial candidate solution” and a relatively quick test of whether it can possibly be completed to a valid solution.
- Thus the 8 queens puzzle can be solved by backtracking. Backtracking is often faster than a brute force strategy.

Problem Solving by Backtracking

Algorithm:

- Start with one queen at the first column 1st row.
- Continue with 2nd queen from the 2nd column 1st row.
- Go up until find a permissible situation.
- Continue with next queen.

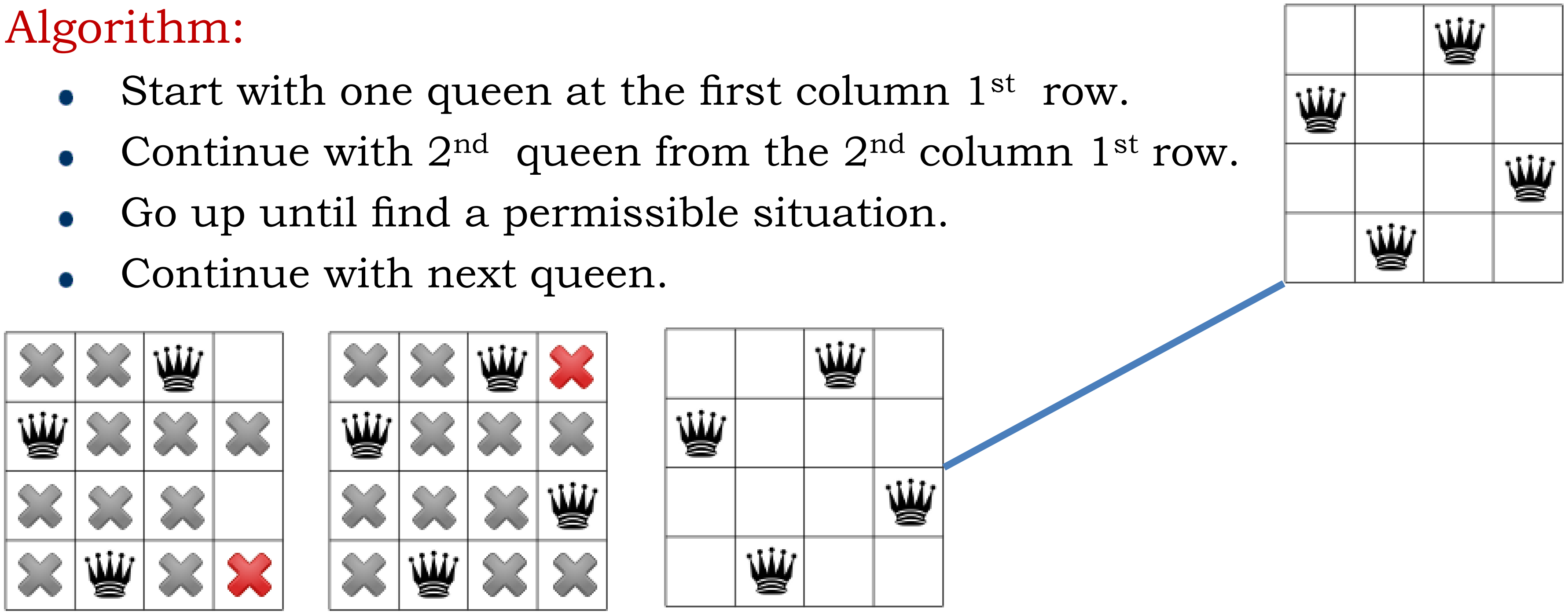




Problem Solving by Backtracking

Algorithm:

- Start with one queen at the first column 1st row.
- Continue with 2nd queen from the 2nd column 1st row.
- Go up until find a permissible situation.
- Continue with next queen.



Happy
Learning