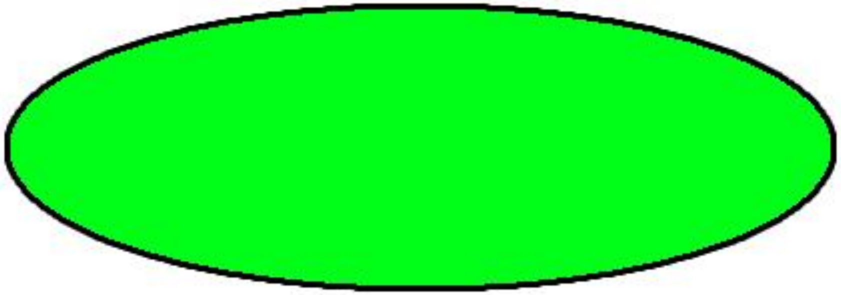
Techniques of Problem Solving

FlowChart

- A flowchart is a type of diagram that represents an algorithm, workflow, or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem.
- The process of drawing a flowchart for an algorithm is known as “flowcharting”.

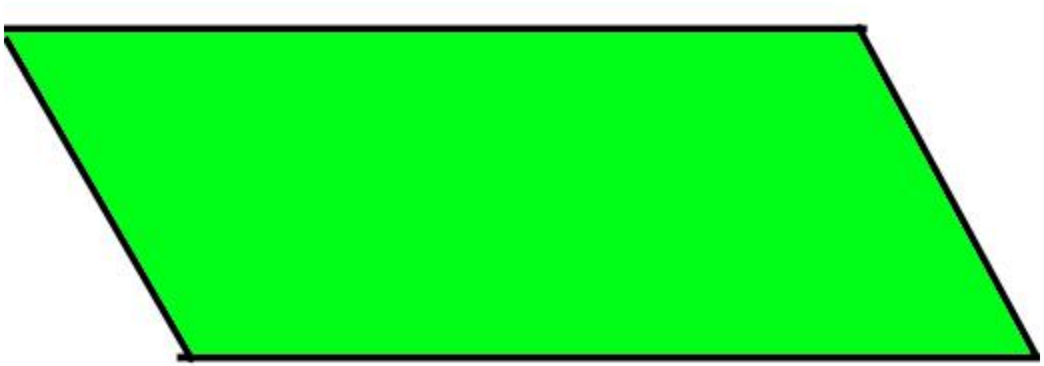
Basic Symbols used in Flowchart Designs

- **Terminal:** The oval symbol indicates Start, Stop and Halt in a program's logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart.



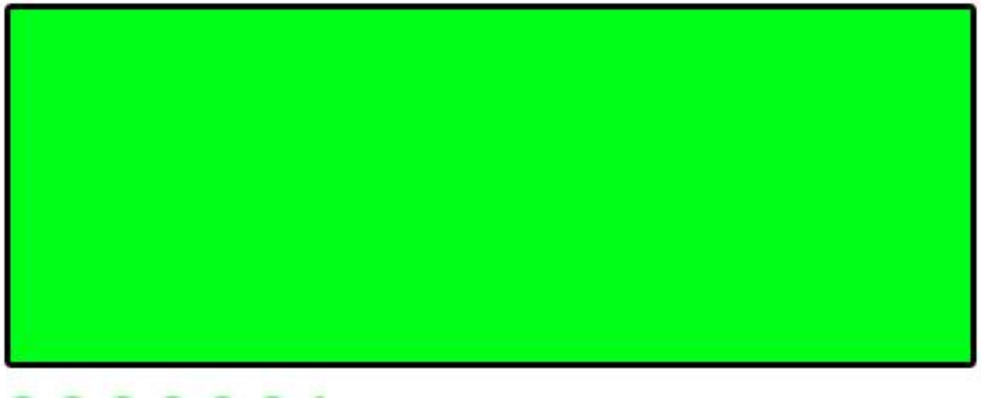
Basic Symbols used in Flowchart Designs

- **Input/Output:** A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



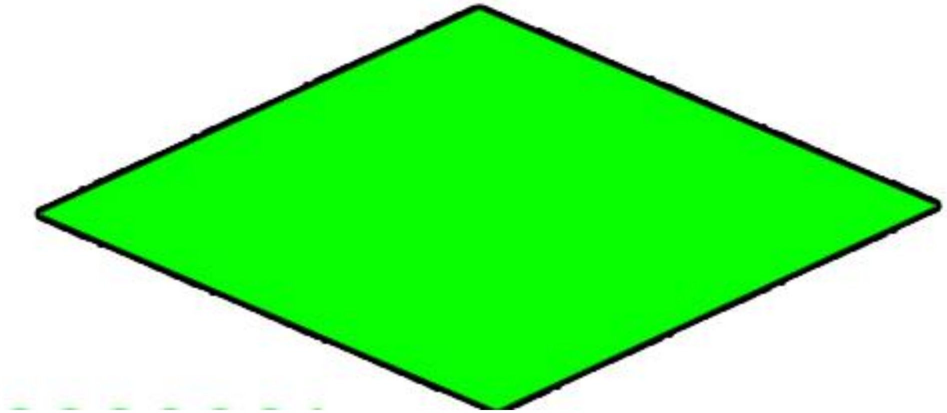
Basic Symbols used in Flowchart Designs

- **Process:** The rectangle depicts a process such as a mathematical computation or a variable assignment.



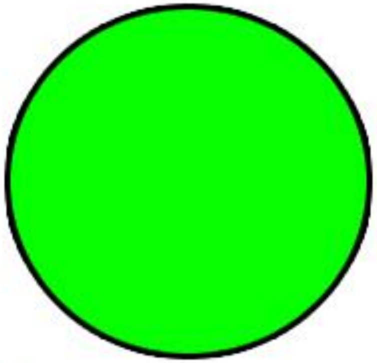
Basic Symbols used in Flowchart Designs

- **Decision:** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



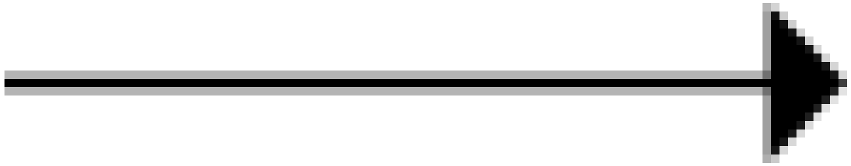
Basic Symbols used in Flowchart Designs

- **Connectors:** Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.

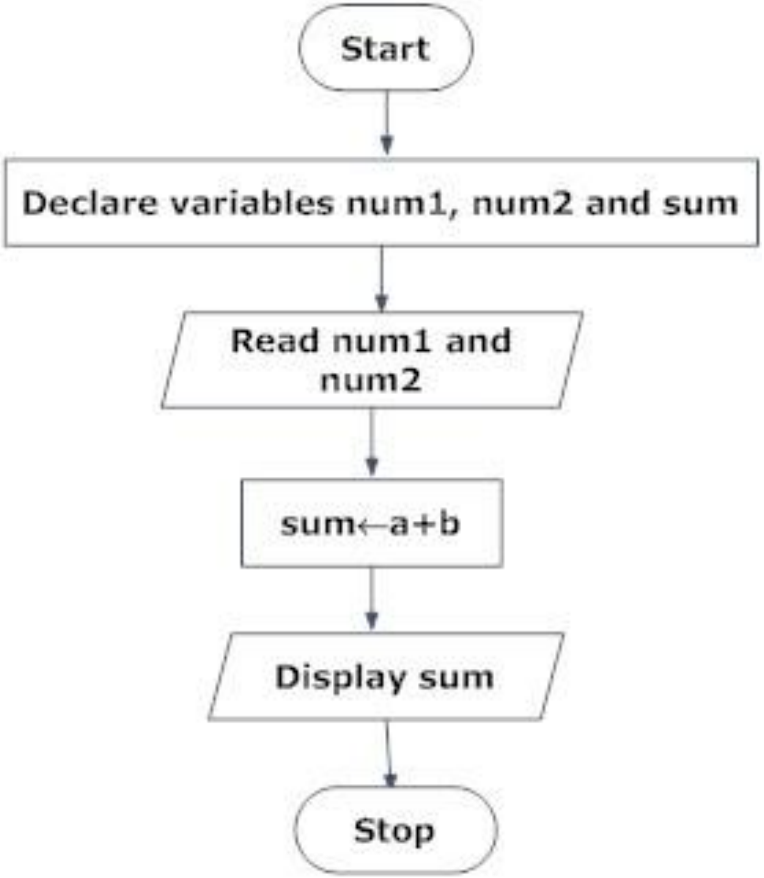


Basic Symbols used in Flowchart Designs

- **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.



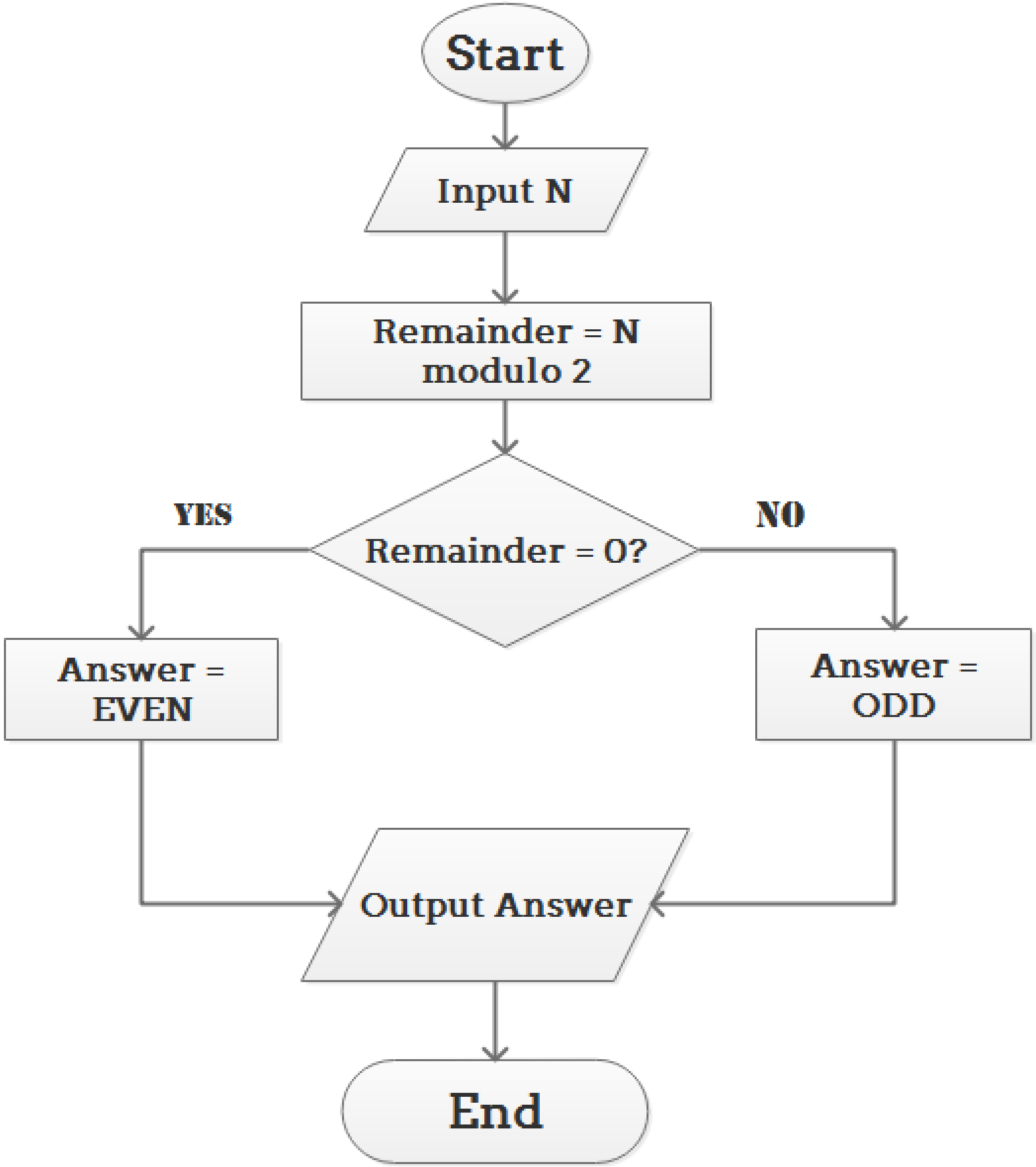
Draw a flowchart to add two numbers entered by user.



Determine and Output Whether Number N is Even or Odd

Algorithm:

- Step 1: Read number N,
- Step 2: Set remainder as N modulo 2,
- Step 3: If remainder is equal to 0 then number N is even, else number N is odd,
- Step 4: Print output.


Start
Input N
Remainder = N modulo 2
Remainder = 0?
YES
NO
Answer = EVEN
Answer = ODD
Output Answer
End

Decision Table

- A decision table is a graphical method for explaining the logic of making decision in tabular format.
- **Decision table** is a brief visual representation for specifying which actions to perform depending on given conditions.
- It shows conditions and actions in a simplified and orderly manner.
- A decision table is similar to a flow chart in use and its construction.
- It may be used either independently or as a complement to a flow chart.

Decision Table

- **The general format of a decision table has four basic parts:**

1. Action entry:

– It indicates the actions to be taken.

2. Condition entry:

– It indicates conditions which are being met or answers the questions in the condition stub.

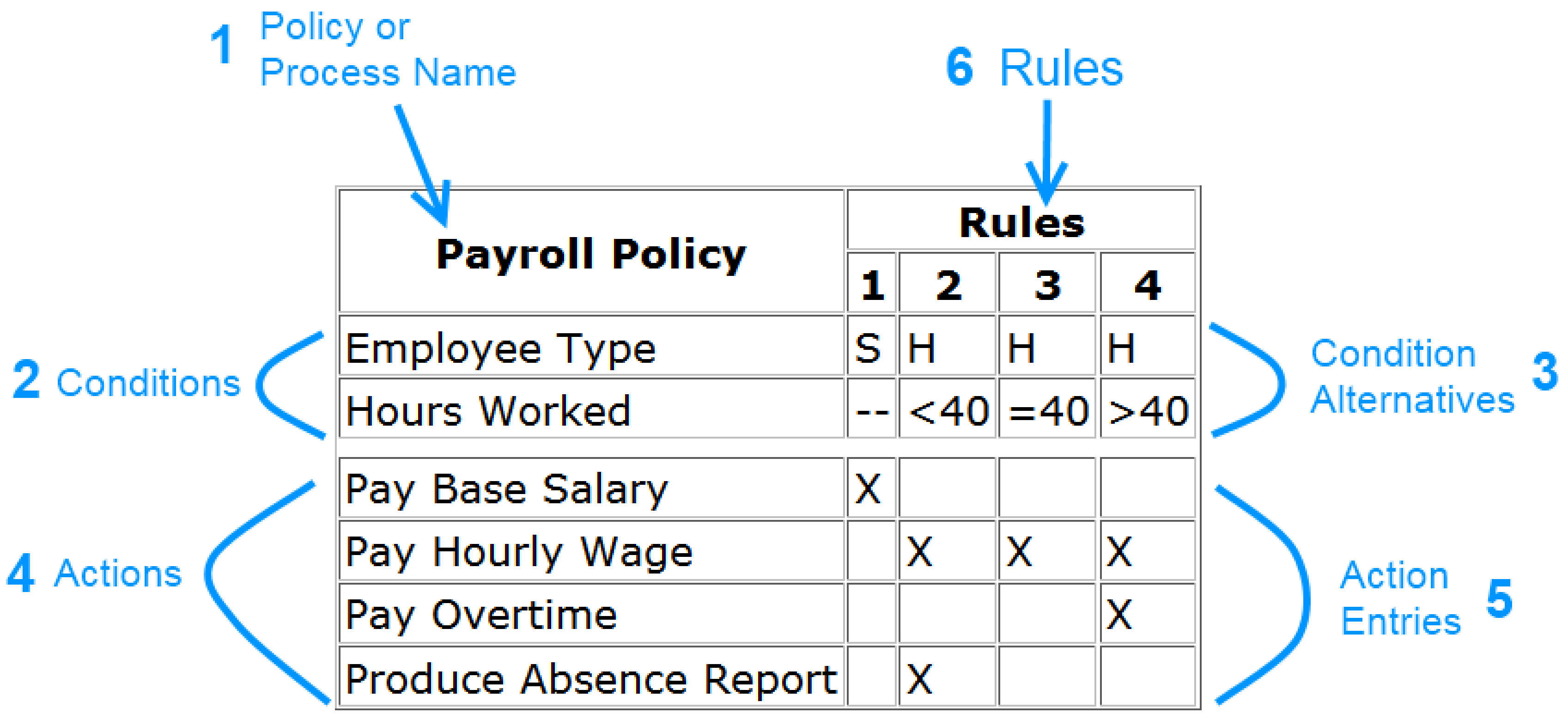
3. Action stub:

– It lists statements describing all actions that can be taken.

4. Condition stub:

– It lists all conditions to be tested for factors necessary for taking a decision.

Decision Table Example



1 Policy or Process Name

6 Rules

2 Conditions

Condition Alternatives 3

4 Actions

Action Entries 5

Payroll Policy	**Rules**			
	1	**2**	**3**	**4**
Employee Type	S	H	H	H
Hours Worked	--	<40	=40	>40
Pay Base Salary	X			
Pay Hourly Wage		X	X	X
Pay Overtime				X
Produce Absence Report		X		

S = Salaried Employee; H = Hourly Employee

Decision Table Example

- For example, Rule 1 states that if an employee is salaried, it doesn't matter how many hours they worked: the action will always be to pay them their base salary. In this example, we are indifferent to the value of the "hours worked" condition - it's value doesn't matter.

Decision Table Example

- **Rule 2:** If an employee is hourly and works less than 40 hours, pay them their hourly wage and produce an absence report for this employee.
- **Rule 3:** If an employee is hourly and works exactly forty hours, pay them their hourly wage.

•

Advantages of Decision table

- It provides compact representation of decision making process.
- Tables are easier to draw up than comparable flow charts. They are easier to change since it is a relatively simple matter to add conditions, rules and actions to a table.
- Tables force the programmer to think the problem through. For example, if there are three conditions to be considered, each of which can be answered yes or no, then there are 2^3 or 8 possible paths or rules.
- The structure of decision table promotes a logically complete and consistent problem definition.

Disadvantages of Decision Table

- It cannot express the complete sequence of operations to solve a problem therefore it may be difficult for the programmer to translate decision table into program.
- If there are too many alternatives, it is difficult to list in decision table.
- It does not show the flow of logic for the solution to a given problem.

Algorithm

- An algorithm is a well-defined step by step solution or a series of instructions to solve a problem.
- Algorithms help to do a task in programming to get the expected output.
- The Algorithm designed are **language-independent**, i.e. they are just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.

Characteristics of an Algorithm

- **Clear and Unambiguous:** Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs**: If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded.

Characteristics of an Algorithm

- **Finiteness:** The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.
- **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon with the available resources. It must not contain some future technology, or anything.
- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

Algorithm

In order to write an algorithm, following things are needed as a pre-requisite:

- The **problem** that is to be solved by this algorithm.
- The **constraints** of the problem that must be considered while solving the problem.
- The **input** to be taken to solve the problem.
- The **output** to be expected when the problem the is solved.
- The **solution** to this problem, in the given constraints.

Then the algorithm is written with the help of above parameters such that it solves the problem.

Example: Algorithm to add 3 numbers and print their sum:

1. START
2. Declare 3 integer variables num1, num2 and num3.
3. Take the three numbers, to be added, as inputs in variables num1, num2, and num3 respectively.
4. Declare an integer variable sum to store the resultant sum of the 3 numbers.
5. Add the 3 numbers and store the result in the variable sum.
6. Print the value of variable sum
7. END

Algorithm, Pseudocode and Program

- **Algorithm : Systematic logical approach** which is a well-defined, step-by-step procedure that allows a computer to solve a problem.

- **Pseudocode :** It is a **simpler version of a programming code** in plain English which uses short phrases to write code for a program before it is implemented in a specific programming language.

- **Program :** It is exact code written for problem **following all the rules of the programming language.**