**# Thought process for solving a problem:**

1. "Understand" the Problem.

2. See the "Given Values" in the problem.

3. Figure out the "Approach" for solving the problem.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**# Algorithm:**

---> A set of finite number of well-defined steps for solving a problem is called an "Algorithm".

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**# Layman Example of an Algorithm:**

@ Algorithm for making a "Cup of Tea":

1. Take 1/2 cup water in a boiling vessel and start boiling it.

2. Add tea leaves and sugar and ginger to the boiling vessel as per your need.

3. Add 1/2 cup milk to the boiling vessel once the colour of water turns brownish.

4. Now boil your tea for few minutes.

5. And after that take your cup and keep a sieve on it and pour your tea inside the cup from the boiling vessel.

6. Your cup of tea is ready to serve/drink.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**# Understanding Human-Computer Interaction:**

\* You can write an algorithm for your approach of solving a problem, and using this algorithm you can make yourself and any other human understand better that how you want to solve a specific problem.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**@ You can represent an algorithm in 2 ways:**

(i) Flowchart

(ii) Pseudocode

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

# But computer doesn't understand "Natural Language", "Flowcharts" and "Pseudocodes".

# We all know that computer only understands Zeroes and Ones (0s and 1s).

# And humans cannot write or speak in 0s and 1s.

@ So, there is a communication gap between humans and computers, and that's why they're not able to communicate with each other.

@ Therefore, for bridging this communication gap between humans and computers we use "High Level Programming Languages".

---> We can code our approach of solving a problem using a high-level programming language, and that programming language will have a tool called compiler/interpreter and it will compile/interpret the code, and due to this compilation/interpretation the code will be converted into a "Machine Understandable Code (0s and 1s)".

---> And then the computer will understand those 0s and 1s, and it will perform that specific task which was coded by the coder.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**# Understanding Flowcharts:**

---> Flowchart is a graphical representation of an Algorithm.

---> The process of drawing a flowchart for an algorithm is known as "Flowcharting".

**@ Basic Flowchart Components:**

\* Ellipse/Pill Shape -> Used for starting or ending our flowchart (Ellipse is aka Terminal/Terminator used for defining [Stop/Start])

\* Parallelogram -> Used for taking "Input or Output" (I/O)

\* Rectangle -> Used for "Processing" (Initialising, or declaring something, or performing calculations)

\* Diamond -> Used for "Decision Making" (It has two outgoing branches "Yes" and "No")

\* Arrows -> Used for explaining the "Flow" (Arrows are aka Flow Lines)

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

@ For understanding **"How to make a flowchart"** and for understanding **"Extra Flowchart Components in depth other than above mentioned 5 basic Components of Flowchart"** refer the below articles:

\* Programiz (Flowchart In Programming):

<https://www.programiz.com/article/flowchart-programming>

\* ConceptDraw (Flowchart Components):

<https://www.conceptdraw.com/How-To-Guide/diagram-software-flowchart-component>

# Note:

---> Every path in a flowchart should be ended using an ellipse which says "end".

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**# Understanding Pseudocode:**

---> Basically, pseudocode is a textual representation of an algorithm.

---> Pseudocode describes how you would implement an algorithm without getting into syntactical details.

\* Pseudo means "fake", and a code means “some logic which is written in a proper syntax of any programming language”.

\* So, a pseudocode is called "fake code" because it is a code which is written in an informal way, informal way means it is written using "natural language" + "basic programming fundamental concepts" like loops and if-else statements, and pseudocode is written without using any programming language's proper syntax.

\* And that's why pseudocode is also called a code which is "programming language independent", and hence it cannot be understood by any computer because it cannot be compiled or interpreted by any compiler/interpreter.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

@ For understanding **"How to write a pseudocode"** refer the below article:

\* GeeksForGeeks (How to write a Pseudo Code?):

<https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>

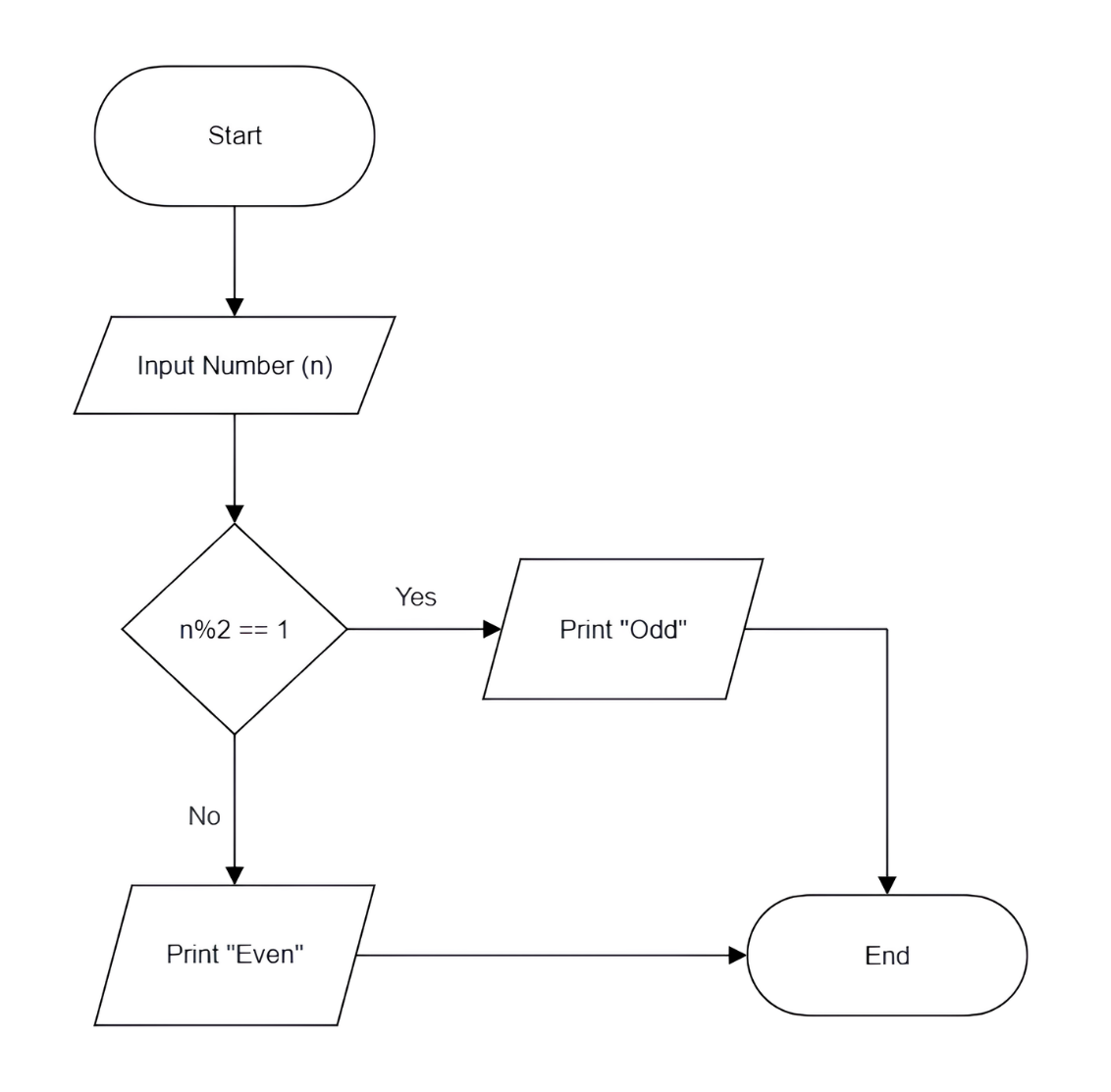
\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**@ Examples of Making Flowcharts and Writing Pseudocodes:**

**\* Ex-1:** Draw a flowchart and write pseudocode for **"checking whether a number is prime or not".**

Ans:

1) Flowchart:



2) Pseudocode:

(i) Read number (n)

(ii) if (n%2==1), then print "Odd"

else print "Even"

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**\* Ex-2:** Draw a flowchart and write pseudocode for **"checking whether a number is positive, or negative, or 0".**

Ans:

1) Flowchart:

A diagram of a flowchart

Description automatically generated

2) Pseudocode:

(i) Read number (n)

(ii) if (n>0), then print "Positive"

(iii) else if (n<0), then print "Negative"

(iv) else print "0"

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**\* Ex-3:** Draw a flowchart and write pseudocode **for "printing counting 1 to n, take input of "n" from user".**

Ans:

1) Flowchart:

A diagram of a flowchart

Description automatically generated

2) Pseudocode:

(i) Input number (n)

(ii) i = 1

(iii) if (i > n), then exit

(iv) else

print i

i = i + 1

go to step (iii)

@ Note:

---> You can also use a loop like for loop or while loop to write above pseudocode.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**\* Ex-4:** Draw a flowchart and write pseudocode for **"taking n input from user and then printing the sum of those numbers".**

Ans:

1) Flowchart:

A diagram of a flowchart

Description automatically generated

2) Pseudocode:

(i) Read number (n)

(ii) i = 1, sum = 0

(iii) if (i > n), then print sum and exit

(iv) else

read number (num)

sum = sum + num

i = i + 1

go to step (iii)

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**\* Ex-5:** Draw a flowchart and write pseudocode for **"printing 1 to n odd numbers only, take input of n from user".**

Ans:

1) Pseudocode:

(i) Input number (n)

(ii) i = 1

(iii) if (i > n), then exit

(iv) else

print i

i = i + 2

go to step (iii)

--->

1) Flowchart:

A diagram of a flowchart

Description automatically generated

**# Note:**

---> Make sure to do a "Dry Run" of your flowchart and pseudocode, so that you're sure that you have written the logic correctly for the given problem.

@ Dry Run:

---> In computer programming, a dry run is a mental run of a computer program, where the computer programmer tests the source code by using variety of inputs to check whether the code is working correctly or not.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**# References Cited:**

\* Programiz (Flowchart in Programming):

<https://www.programiz.com/article/flowchart-programming>

\* ConceptDraw (Flowchart Components):

<https://www.conceptdraw.com/How-To-Guide/diagram-software-flowchart-component>

\* GeeksForGeeks (How to write a Pseudo Code?):

<https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_