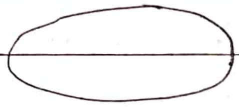
flowchart & pseudocode

flowchart :-

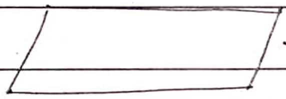
visualization of our thought process or algorithm and represent them diagramatic-ally is called flow chart.

Symbols :-



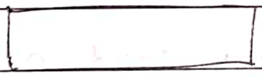
→ start/ stop → indicate starting & ending of flow chart

oval



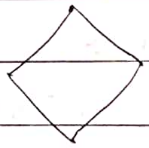
→ Input/ output → indicate input & output in flowchart.

parallelogram



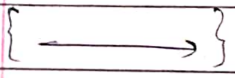
→ process ing → indicate process like mathematical computation (increment/decrement) or variable assignment.

rectangle



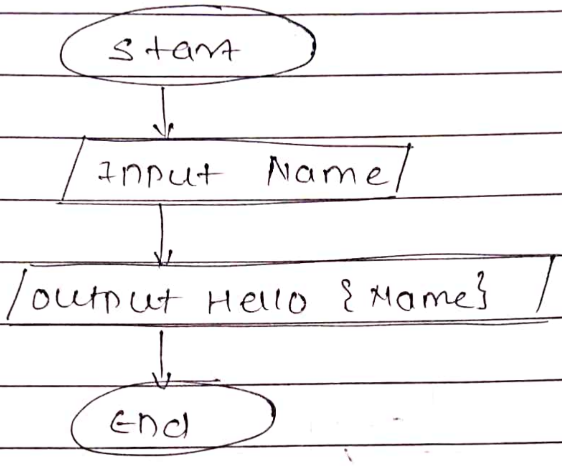
→ condition → conditional statement which results in true/false (yes/no).

Diamond

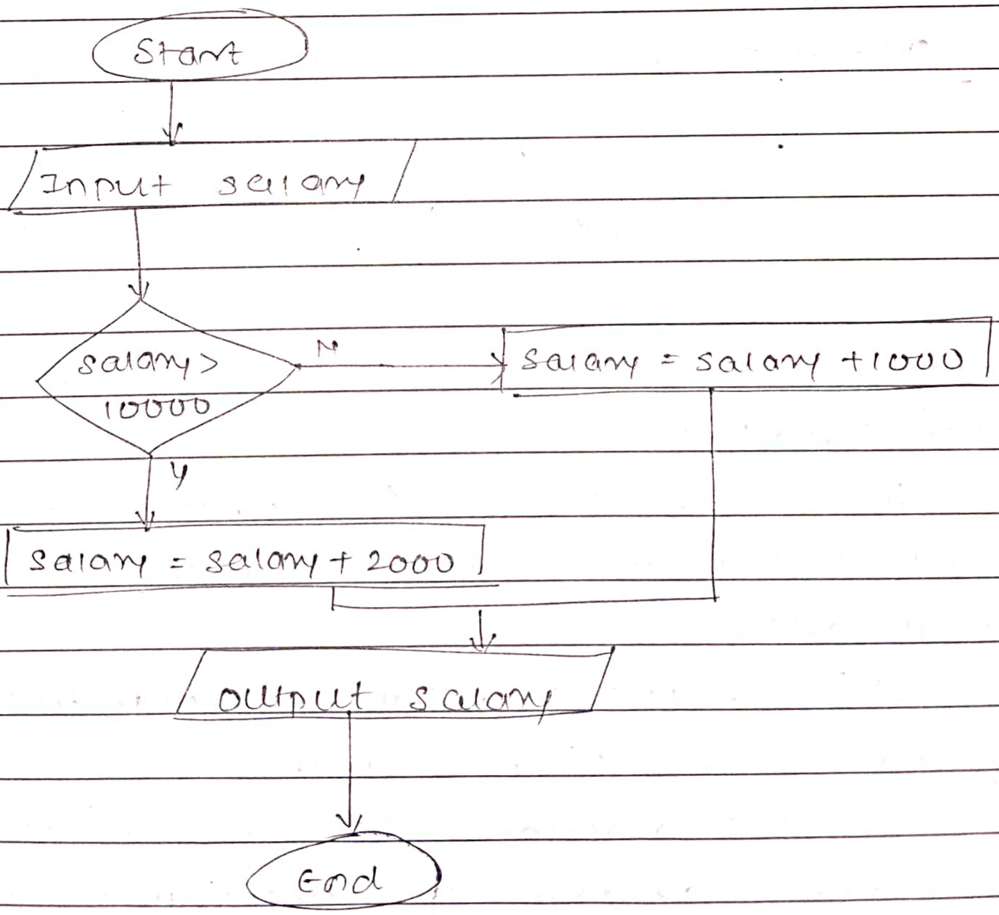


→ flow ~~of~~ direction of program → indicate flow of the program.

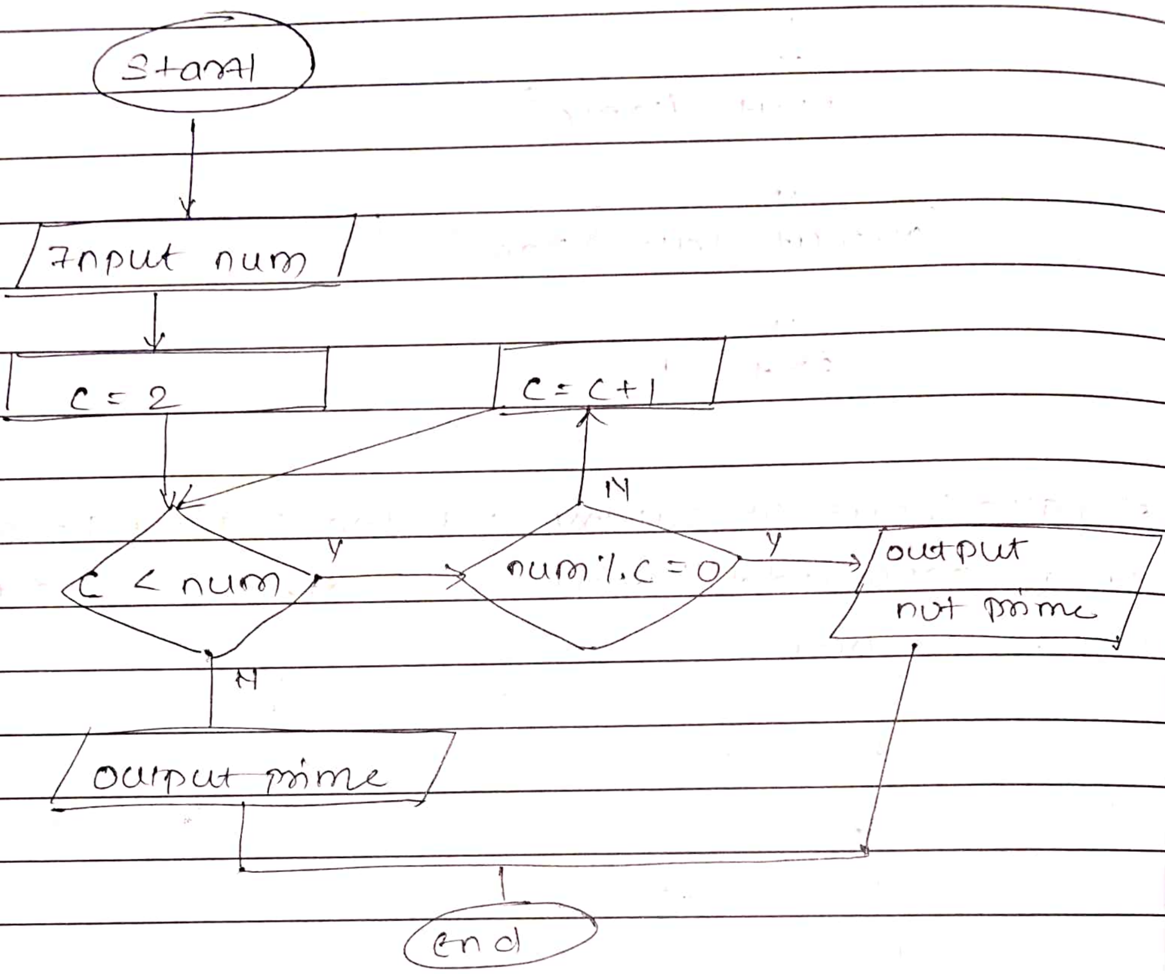
e.g. 1] Take {name} and output Hello {name}.



2) Take input of salary. If the salary is greater than 10000 add bonus 2000, otherwise add bonus as 1000.



3) Input a number & print whether it is prime or not.



* Pseudocode :

It is just a way to write steps which is human readable format [It is not a code].

It is mainly ~~#~~ meant for human reading not for machine reading.

- It is like a rough code which shows how the algorithm of a program works.
- It does not requires syntax.

e.g. 1] pseudocode of e.g.(2) of flowchart

```
start
Input salary
if salary > 10000 :
        salary = salary + 2000
else :
        salary = salary +1000
output salary
end.
Exit.
```

2] pseudocode of prime no. or not

```
start
Input num
if n≤1 :
      print "neither prime nor composite".
c=2
while c < num:
  if num % c==0 :
        print "not prime"
        exit
   c=c+1
end while
print "prime "
Exit
```

Optimization of prime soln:

let's have a no. to check prime /not → 36

$$\left[\begin{array}{lll} 1 \times 36 & = & 36 \\ 2 \times 18 & = & 36 \\ 3 \times 12 & = & 36 \\ 4 \times 9 & = & 36 \end{array}\right] \longrightarrow (1)$$

$$6 \times 6 = 36$$

$$\left[\begin{array}{lll} 9 \times 4 & = & 36 \\ 12 \times 3 & = & 36 \\ 18 \times 2 & = & 36 \\ 36 \times 1 & = & 36 \end{array}\right] \longrightarrow (2)$$

In above demonstration we have clearly seen that (1) & (2) are repeated, so to optimize this we can ignore (2)

As same as this we can check no. is prime /not by travelling from 2 to $\sqrt{num}$.

e.g. To check 17 is prime /not, we do not have to travel from 2 to 17, we just have to travel from 2 to $\sqrt{17}$ (i.e 4).

```
start
Input num
if n ≤ 1 :
        print "neither prime nor composite"
c = 2
while c*c <= num :
```

```
if num % c ==0:
    output "not prime"
    exit
c= c+1
End while

output "prime".
Exit.
```

example: Input 2 no. & print sum:

```
Start
Input num1, num2
sum = num1 + num2
output ~~e~~ sum
Exit
```