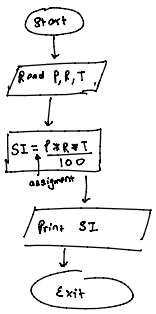
Flowchart

12 September 2021 12:09

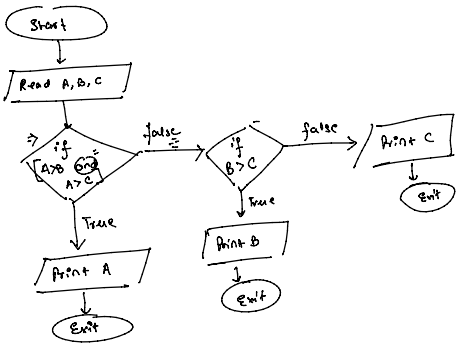
↳ Components of a flowchart

1. ⬭ ← start / end
2. ▱ ← read / write
3. ▭ ← processing
4. ◇ ← conditional logic
5. ○ ← connector
6. ↓↑

1. Simple Interest



2. find largest of 3 nos.



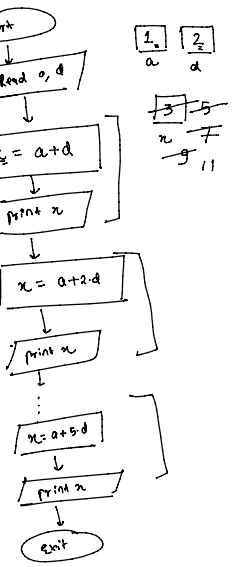
3. a, d

a+d, a+2·d, a+3·d, a+4·d, a+5·d

a = 1, d = 2 ← logic case

3, 5, 7, 9, 11

o/p: 3, 5, 7, 9, 11



a + 1·d

a + 2·d

a + 3·d

⋮

a + 5·d

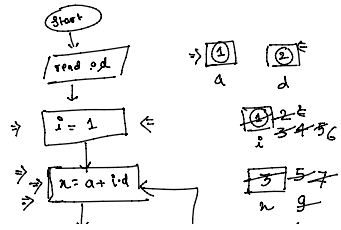
x = a + i·d

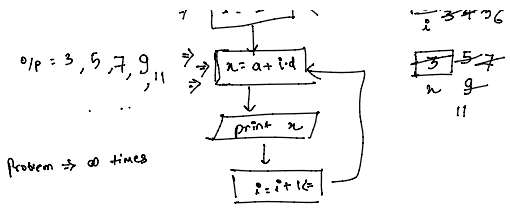
1 ≤ i ≤ 5

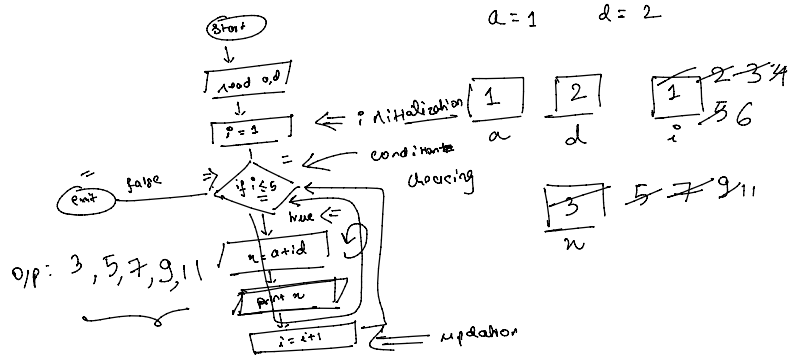
Simply

a = 1 d = 2

o/p = 3, 5, 7, 9, 11



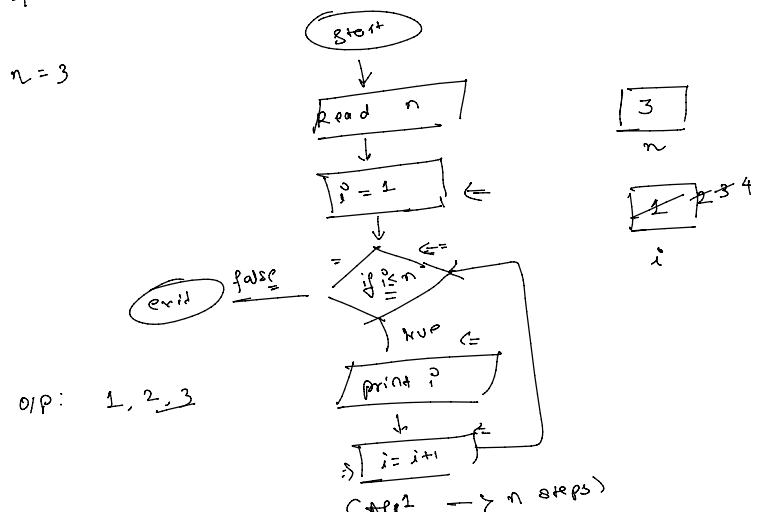




1. Initialisation ×
2. condition check ×
3. Body (optional)
4. update ×

4. Print nos from 1 to n

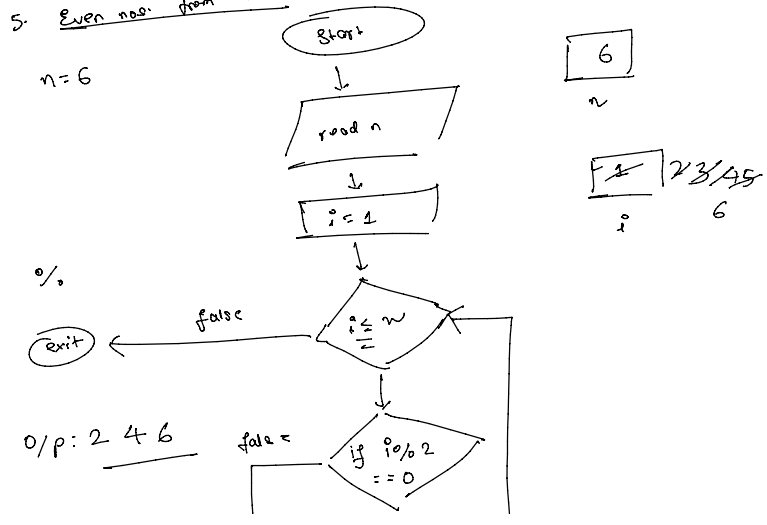
n = 3



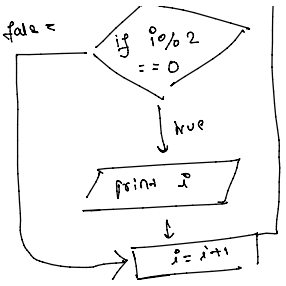
(Appl → n steps)

5. Even nos. from 1 to n

n = 6

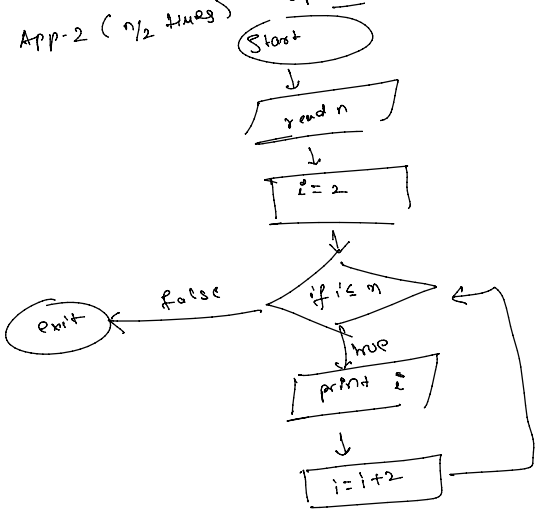


O/P: 2 4 6

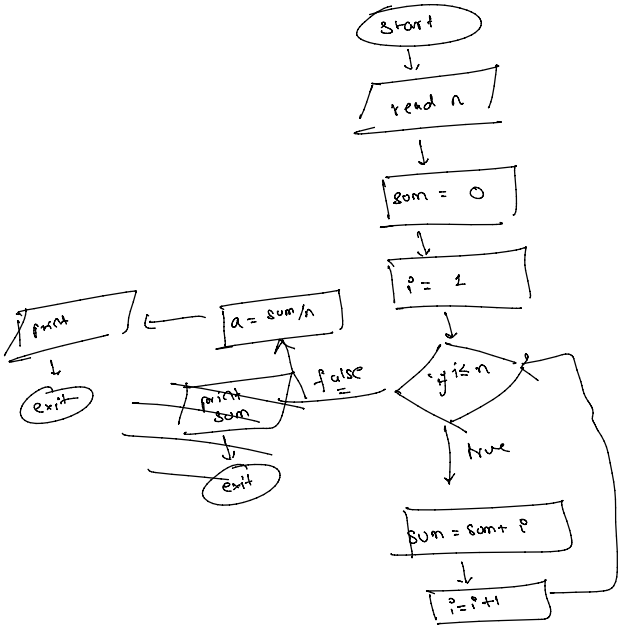


App-2 (n/2 times)

Optimal



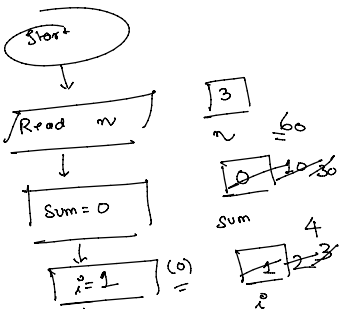
6. Given a no. n, calculate the sum. of nos. from 1 to n.

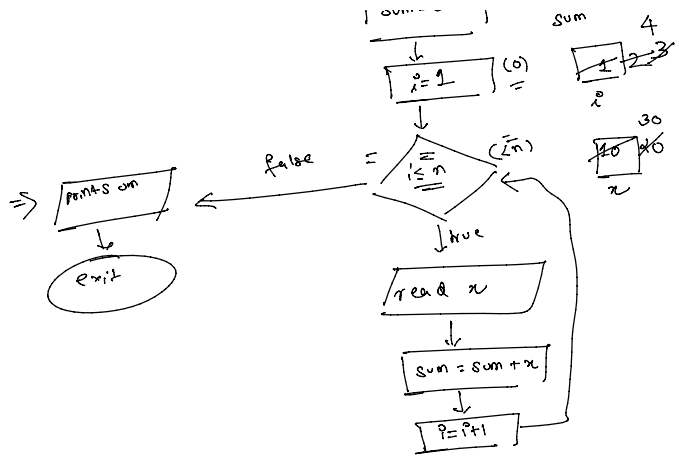


↳ Reading n values and calculating the sum

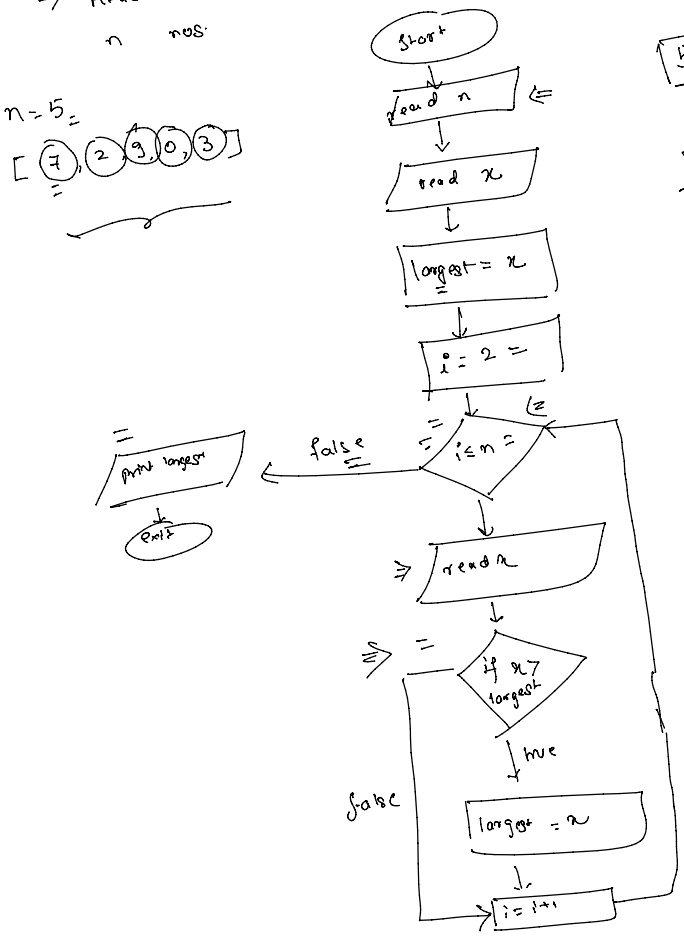
n = 3

[10, 20, 30]





↳ Read n nos. and find the largest of n nos.



n = 5

[7, 2, 9, 0, 3]

n = 5 ⇒ i = 2 ⇒ i = 1 ⇒ i = 0

largest = 7 → 9

$\leq n \Rightarrow n-1$

$\leq n-1 \Rightarrow n-1$

or

$\leq n \Rightarrow n-1$

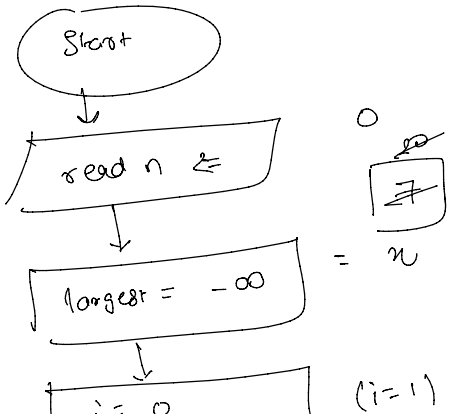
$< n-1$

$\leq n-2$

2 3 4 5 6

n = 3

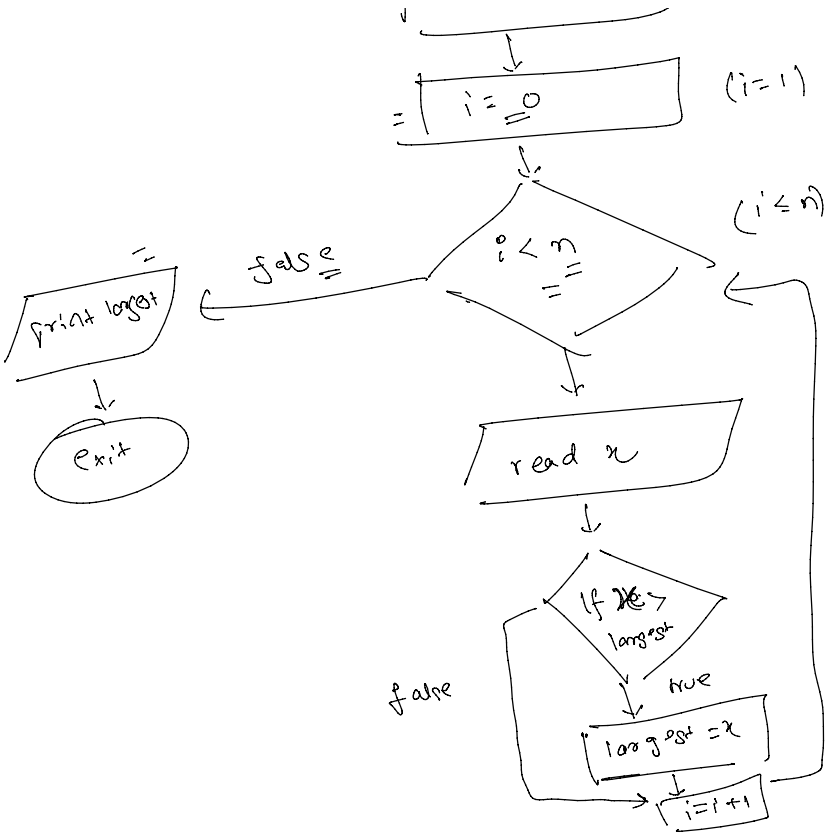
[7, 20, 0]



n = 3

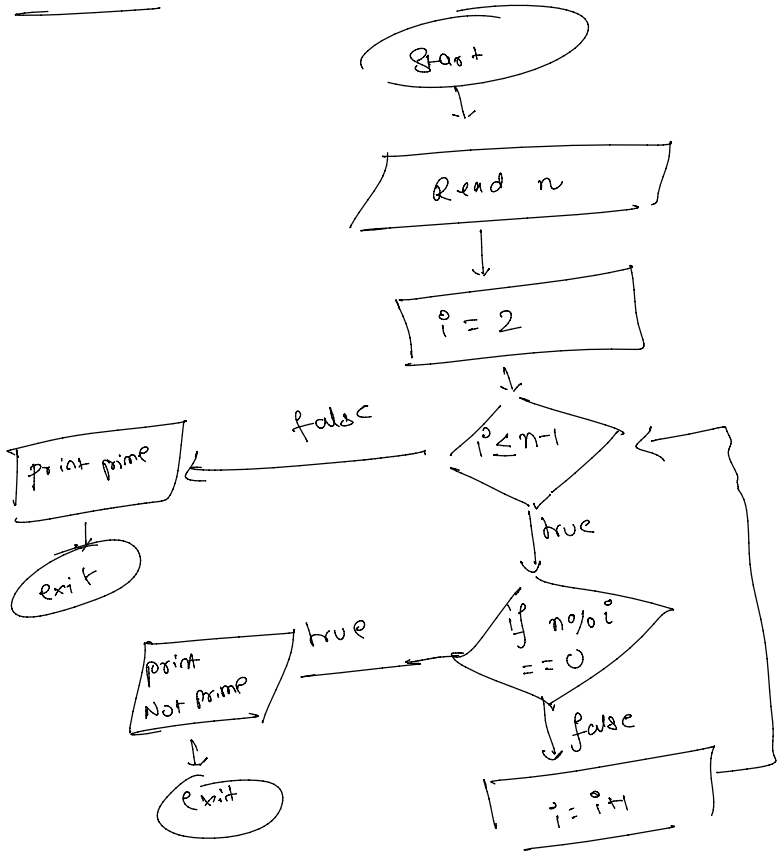
largest = -∞ → 7 → 20

i = 0 → 2 → 3



0 + 2
i 3

↳ Prime no.



Pseudo code

- read n
- print n
- i ← 0 (assignment)
- if [condn] then ... else then ... endif
  (if [condn] then ... endif)
- while (condn) then
  ....
  endwhile

read n

i ← 2

while (i ≤ n-1) do

⇒ if (n %% i == 0) then

print n is not prime

exit

else then

i = i + 1

endif

endwhile

⇒ print n is a prime

exit

n = 5

n: 5 4

i: 2 3 4 5

n = 4

for each row

print n '*'

row1 * * * *

row2 * * * *

row3 * * * *

row4 * * * * n×n

→ [j ← 1

row 4 * * * * n×n

⇒ read n

⇒ i ← 1

⇒ while i ≤ n do

[print n *]

i ← i+1

⇒ endwhile

⇒ [j ← 1
while j ≤ n do
print *
j ← j+1
endwhile]

n = 3

read n

i ← 1

⇒ while i ≤ n do

j ← 1

⇒ while j ≤ n do

print *

j ← j+1

endwhile

print "n"

i ← i+1

endwhile

* * *
* * *
* * *

3
n

1 2 3 4
i

1 2 3 4
j 1 2 3 4
1

n = 5

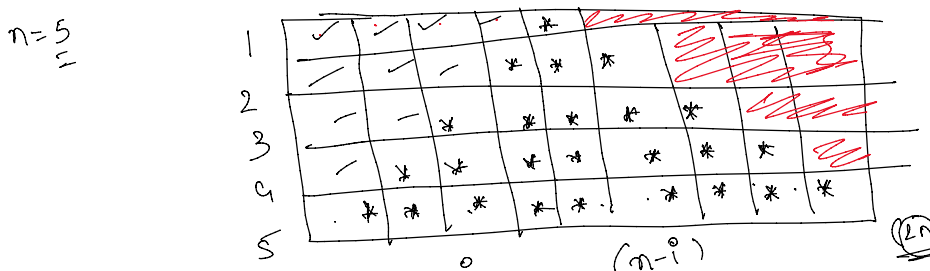
row 1 * 1

row 2 * * 2

row 3 * * 3
*

*
*

*
* * * *



row 1 ⇒ (n-i) spaces
⇒ 4 spaces
⇒ 3 spaces
⇒ 2 spaces
⇒ 1 space
⇒ 0 spaces

foll. by (2n-1) *'s
2i - 1

row 1 ⇒ 2
⇒ 3
⇒ 4
⇒ 5

```
read n
i ← 1
while i ≤ n do
    j ← 1
    while j ≤ n-i do
        print " "
        j ← j+1
    endwhile
    j ← 1
    while j ≤ 2i-1 do
        print *
        j ← j+1
    endwhile
    print "\n"
    i ← i+1
endwhile
```

n = 4