__/__/____

LEC-1

How to approach problem

1. Understan the problem
2. Enter Appropriate input values
3. Create Logics / Algorithm

Problem Statement → Solution (Brain) → Rough Soln ① flow chart ② Psuedo code → HLL (C++) → Machine level language (Binary)

Compiler

① flow chart ?
↳ Diagramatic representation of an algorithm

Component of flow chart

① Start / end

② Read & Print
Input } Data
Output

③ i = i + 1
Main logic

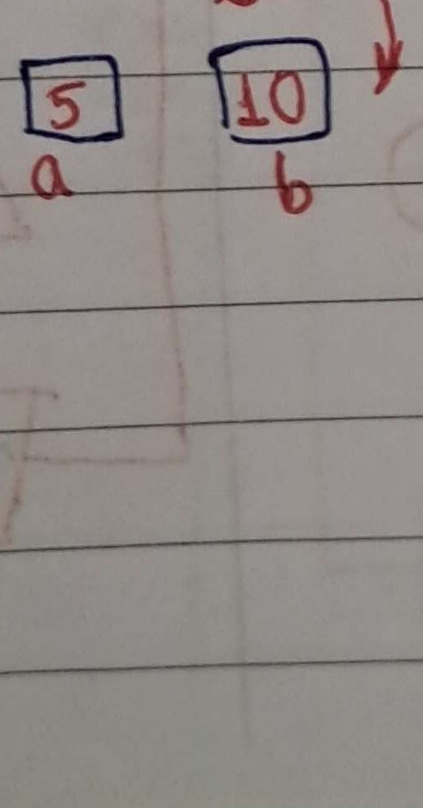
④ is i=0? Yes / NO
Decision Making
→ Conditions

⑤ ↓ ↑
arrow
flow depict

⑥ A
Connector
↳ In function

Sum of 2 no.s

memory block



Pseudocode → Generic way of representing Code

① Read 2 no.s a and b
② Sum a = a + b
③ Print Sum

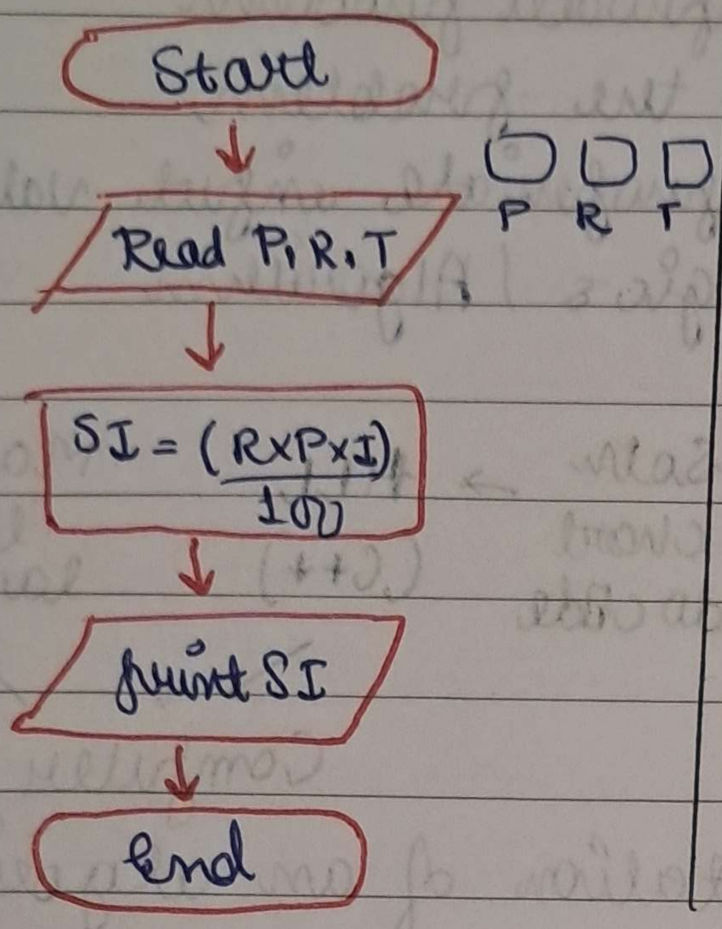
(24)

① Read a
② Read b
③ let sum = 0
④ sum = a + b
⑤ Print sum

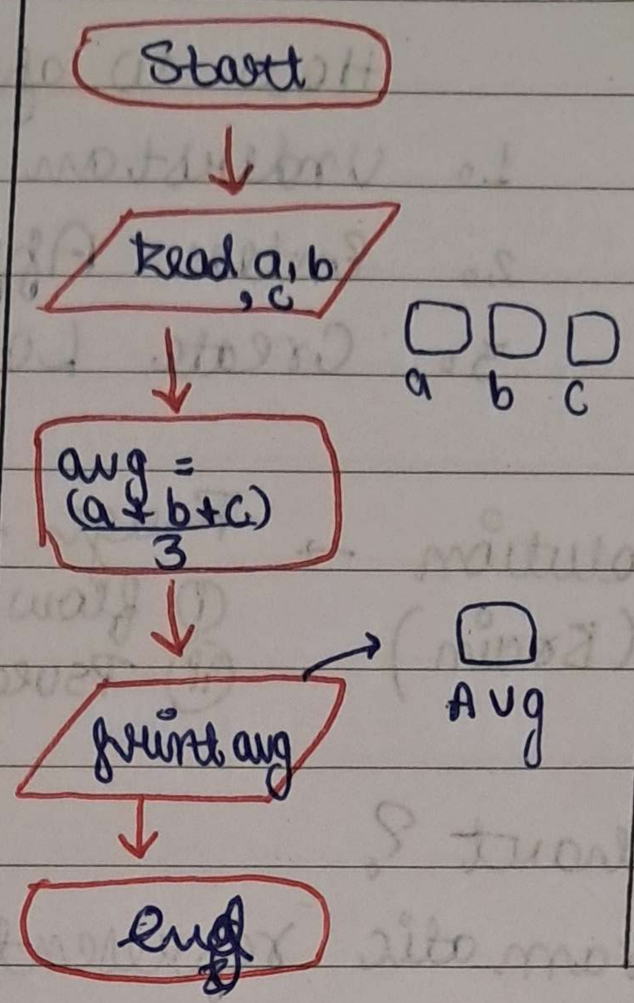
} vary person to person

② Simple Interest

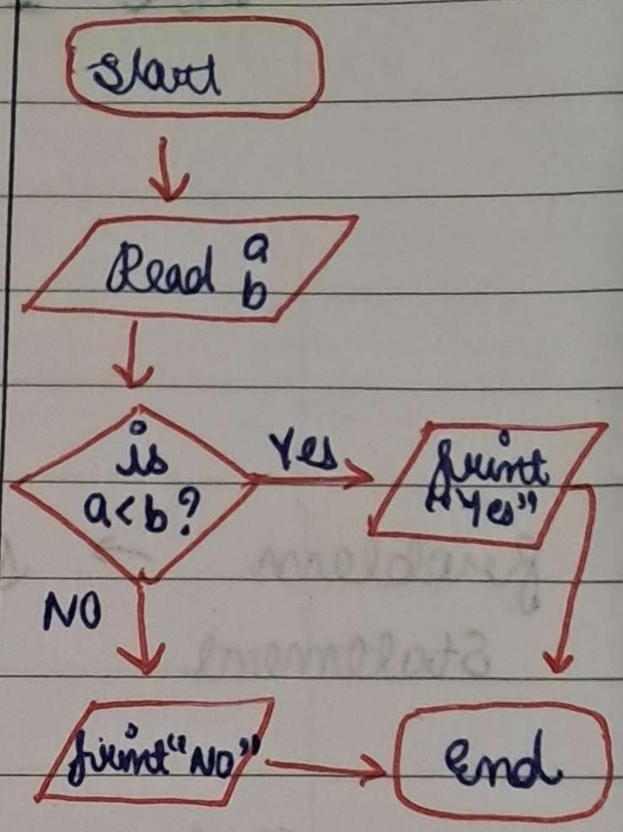
$$S.I = \frac{R \times P \times I}{100}$$



③ Avg of 3 nos.

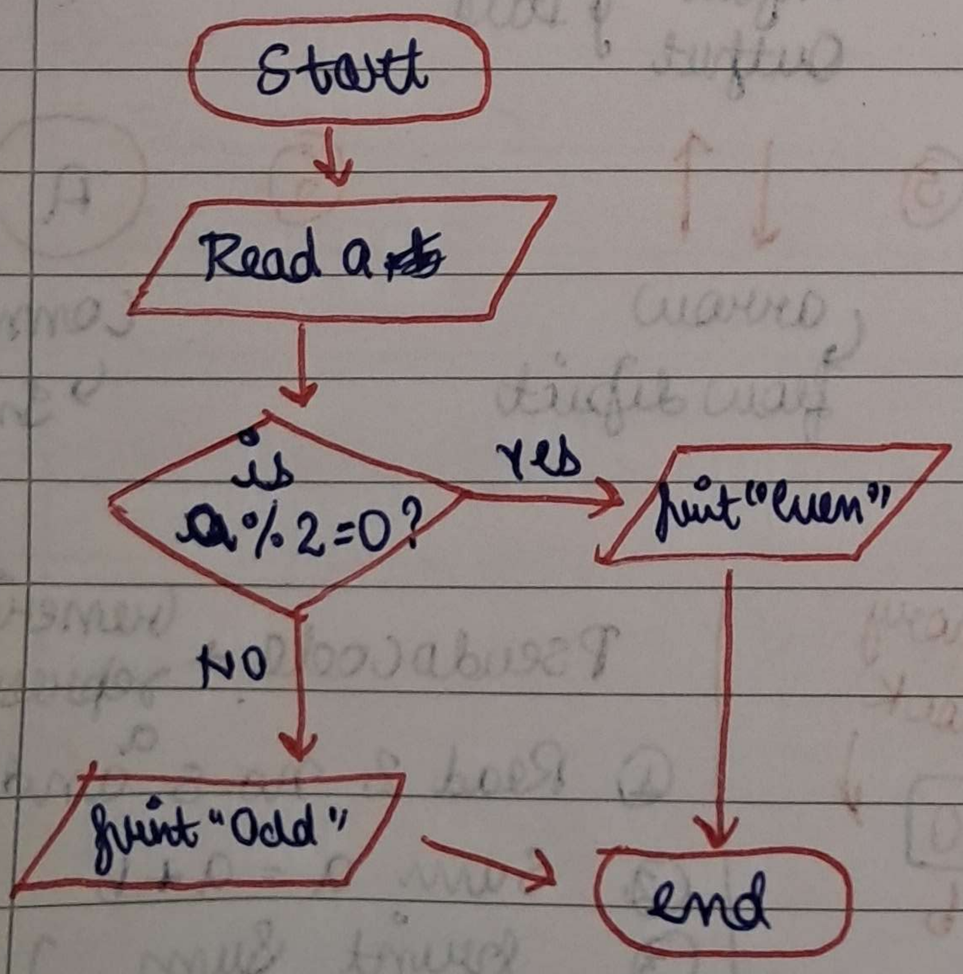


④ a<b? "Yes"/"No"



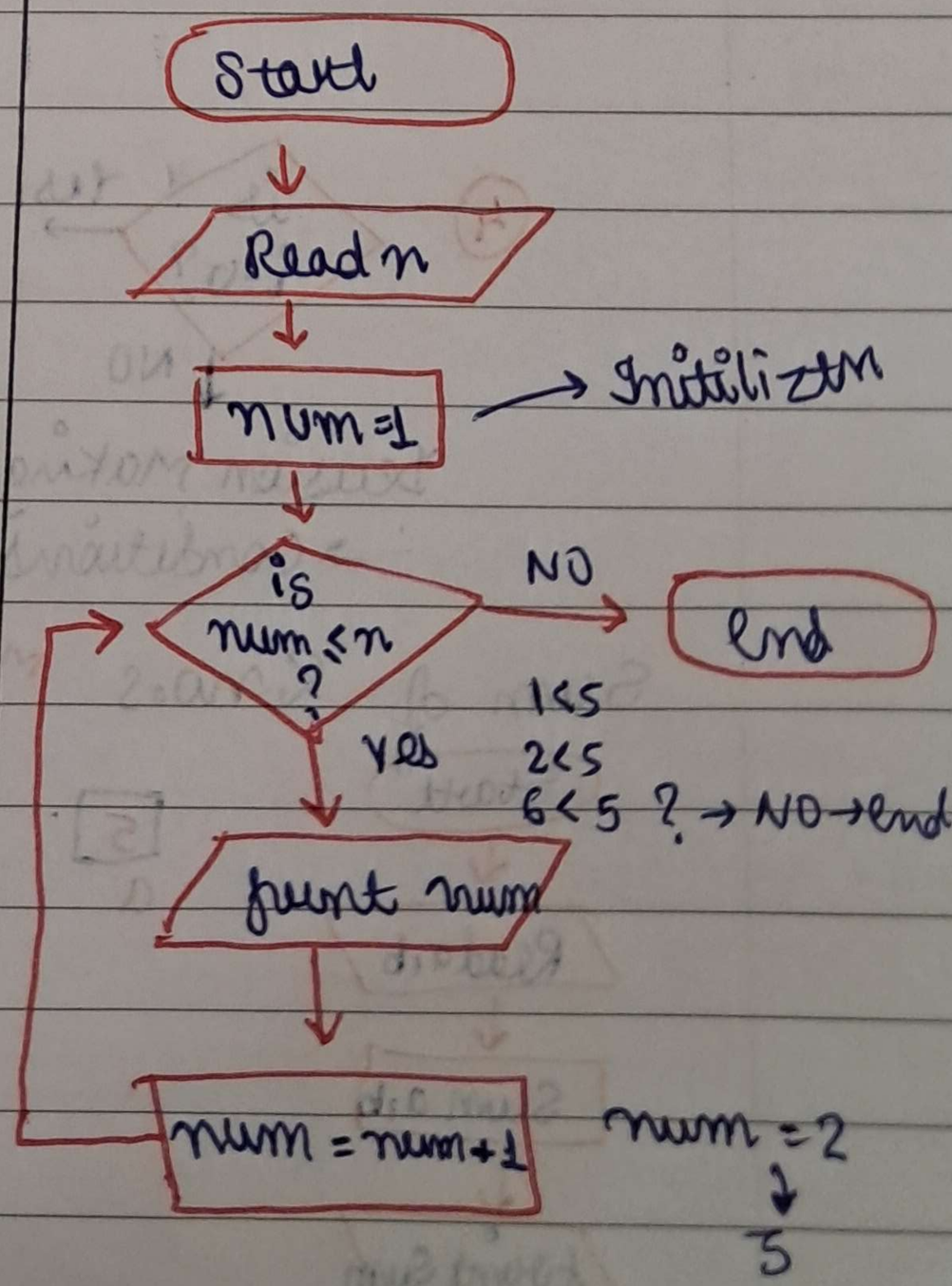
⑤ Odd or even??

n % 2 = 0 → EVEN
n % 2 = 1 → ODD



⑥ Print 1 to N

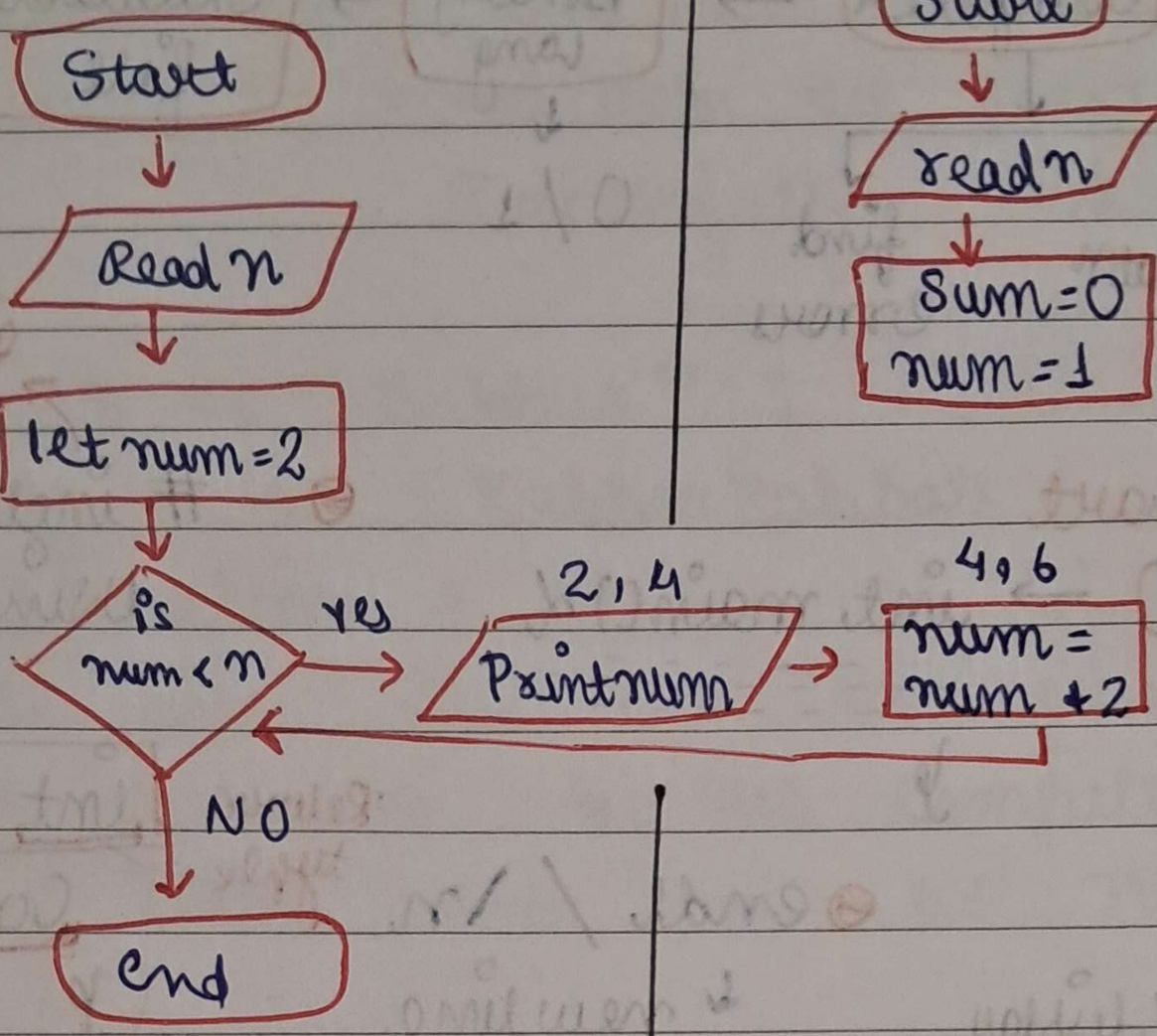
i/p → N = 5
o/p → 1, 2, 3, 4, 5



⑥ 1 to n (evenprint)

i/p → n=5

o/p → 2, 4



⑦ find Sum 1 to n

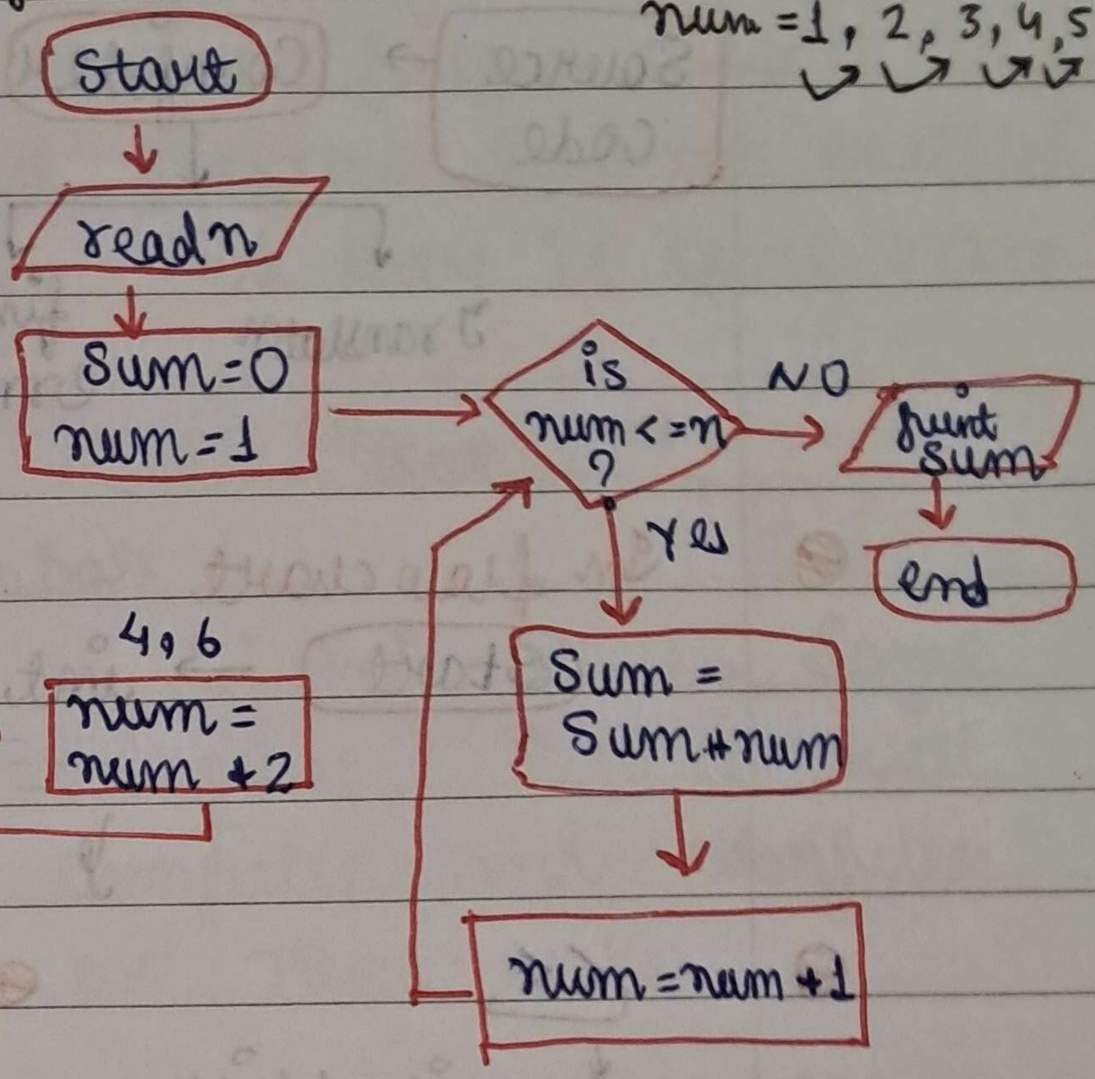
n=5

o/p → 1+2+3+4+5 = 15

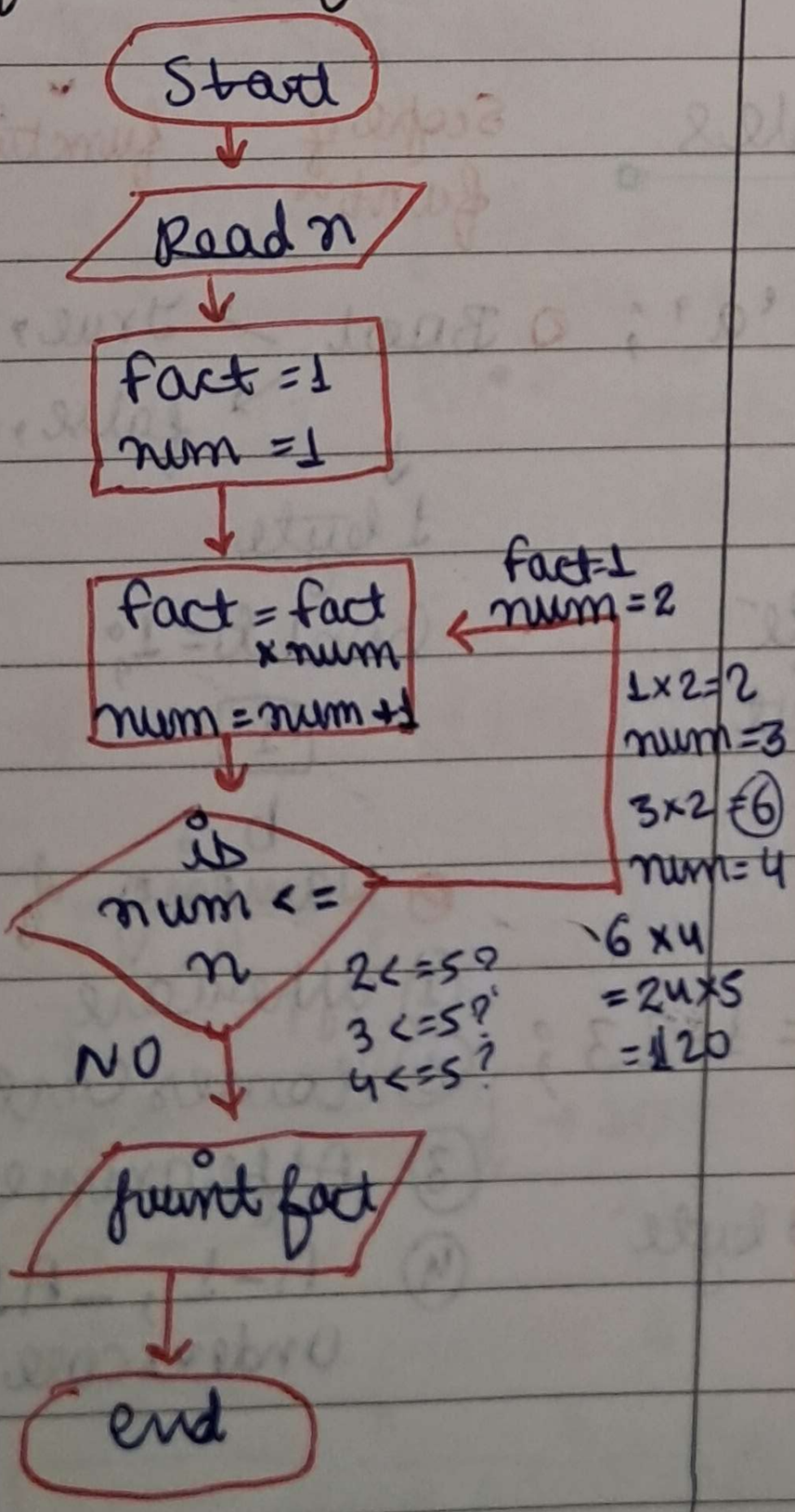
n=5

Sum=0, 1, 3, 6, 10

num=1, 2, 3, 4, 5



⑧ factorial of n



⑨ Check prime or not?

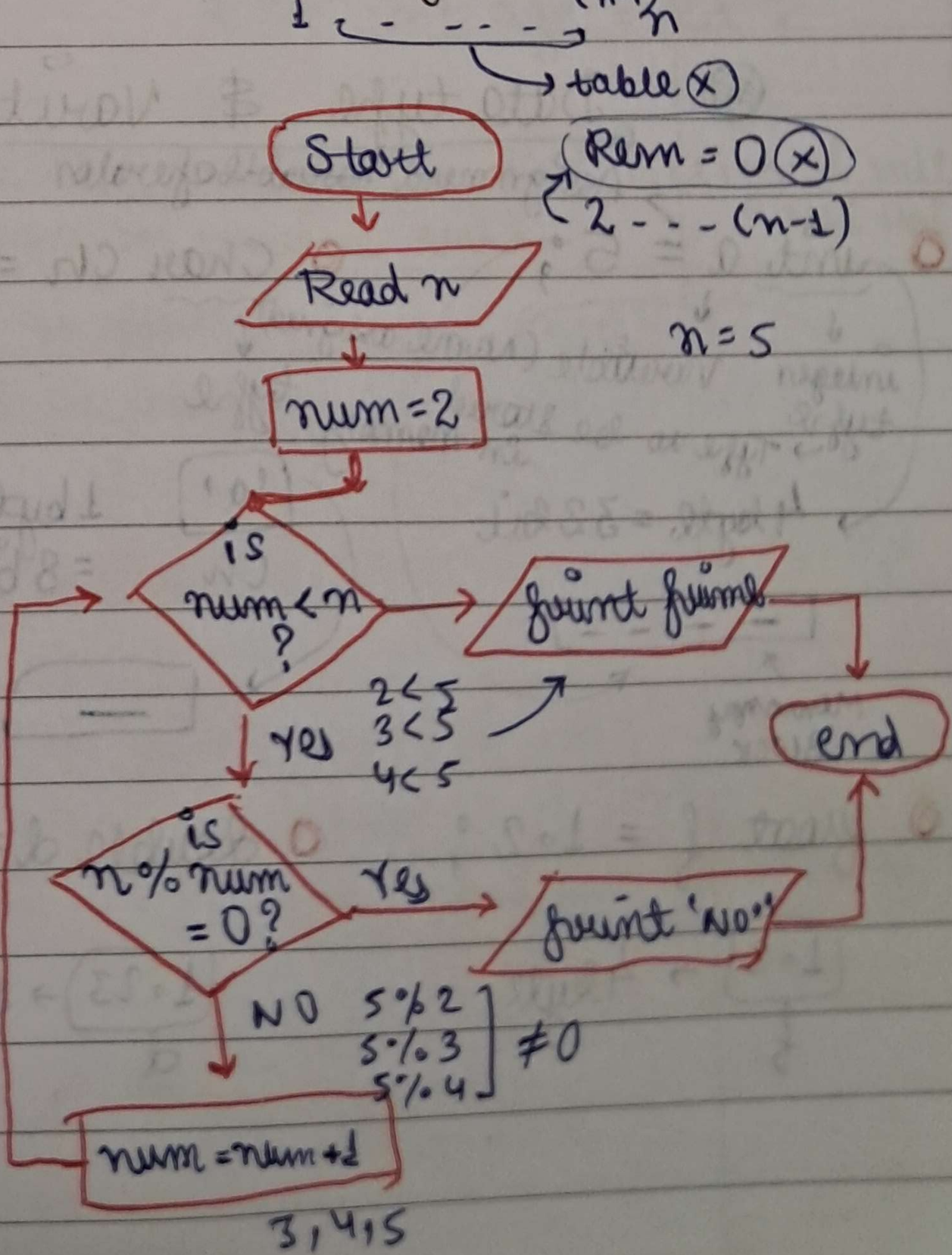
1 - - - - (n-1) n

table ✗

Rem = 0 ✗

2 - - - (n-1)

n=5



__/__/____

LEC-2

Source code → Compiler → Binary lang → Executable file → Computer Run

Compiler → Translation, find errors

Binary lang → 0/1

- In flowchart

Start → int main() {

}

- #include <iostream>
  using namespace std;

→ function

Return type → int main() {

- <<
  ↓ to write/display something

- endl / \n
  ↓ new line

- cout << "" ;

- ; → line khatm krne ke liye

- { → Scope of funtn

- main() → function name

- print to bhi "" mai ho

Data type & Variables

Assignment ~~Variable~~ operator

- int a = 5;
  - int → integer type
  - a → Variable (Name assigned)
  - 5 → Type to be stored in memory
  - 4 byte = 32 bit

- Char ch = 'a';
  - Char → type
  - 'a' → ch
  - 1 byte = 8 bit

- Bool → True, 1 / false, 0
  - 1 byte
  - bool b = 1;
  - 1 → b

Memory Block

- float f = 1.2;
  - 1.2 → 4 byte
  - f

- double d = 1.23;
  - 1.23 → 8 byte
  - d

- Naming of Variable
  1. upperCase
  2. lowerCase
  3. Alphanumerical → abc1 ✓, 1abc ✗
  4. A-1, _A1
     Underscore

→ // or Ctrl + / :- used to Comment
line of code ignore By compiler

→ return 0; → This mean that the program executed Successfully

→ return 1; → This mean the program does not executed Successfully and there is some error.

→ cout << "Babbar"; cout<<"Babbar";
Output :- BabbarBabbar
Explanation → If we use ';' and write some code just after the Semicolon, then it will get executed by the compiler and compiler will treats that code as next task

→ Declaration :-
Ex - int age;

→ Initialisation :-
Ex- int age = 21;

If we print a variable → Jiske kai assigned val Nhi Hai
garbage → random unique val

Ex→ int random;
cout << random;

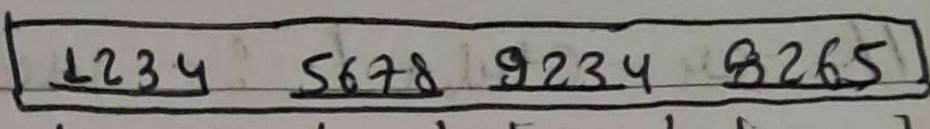
output
43426387
→ unique Always

How data is stored

① int a = 8
4 byte
Binary Convert → 1000
4 bit
8 8 8 8
0000 0000 0000 1000
32 bit
Last 4 bit = 1000
Rest 28 bit = 0000---

② Char ch = 'a' → ASCII Table
1 bit
97 → Binary
8 bit

- Suppose if

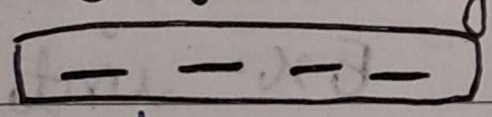


Type Casting

- int a = 'a';
  cout << a << endl;
  O/P → 97
  Since a is mapped to 97 in ASCII table

- char ch = 98;
  O/P → b
  int typecasted into char

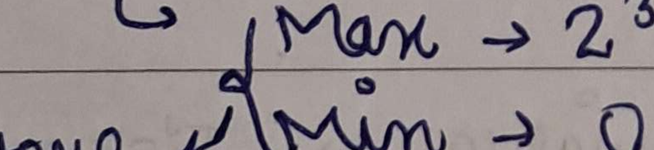
- If Int → 4 byte = 32bit



↳ Max → $2^{32}-1$
Store ↙ Min → 0

Similarly Char → 1 byte = 8 bit
↳ Max → 2^8-1
Min → 0

If we have integer of 2^{16} → less than $2^{32}-1$ But greater 2^8-1 then use store krne Mai



- char ch = 12345;
  cout << ch << endl;
  O/P → @

→ 64 Bit jayega

last value check krke use small 2^8-1 form Mai Bna liya jayega aur output milega

Explicit Typecasting

```
float a = 5.466;
int b = int(a);
cout << b;
```

↳ o/p → 5

How -ve number are stored ??

first bit

+ve → 0 ; -ve → 1

-ve no. → (-5) → Store

1. Ignore the -ve sign → 5
2. Convert into binary repres^n

00	00	00	101

3. Take 2's compliment & store

00000	0000	0000	0000101
11111	1111	1111	11010
			+1
1	1111	1111	11011

1's Compliment / Bit exchange

-ve

```
int a = -5;
print a ??
```

4 byte: 11111111 11111111 11111111 11111011

2's comp: 00000 0000 0000 000100

+1

1's comp: 0000 0000 0000 000101

5

-ve sign

Max bits $= 2^{31}-1$

31 → | 0 | - - - - - - - - - - - - - - - |

→ Rest bits will be considered (+ve)

| 1 | - - - - - - - - - - - - - - - |

→ Rest bits will be considered (-ve)

↳ $-(2^{31}-1)$

| 0 | 0 - - - - - - - - - - - - - - |

→ 0

| 1 | 0 - - - - - - - - - - - - - - |

→ -0 → 0

1 no. → "0"

+

↓

2 Representation use to display

↓

So same one rep ⟶ 2's complement

$-(2^{31}-1)$ - - - - - $2^{31}-1$ } 1 Rep same

↓ to ↓

-2^{31} - - - - - $2^{31}-1$

- Signed data
- unsigned data
- unsigned int a1 = -112;
  print a1
  ↳ Huge value

} Detail in VS code

OPERATORS

① Arithmetic

\+ − * / %

② Relational

= > < >= <= !=

int / int = int

③ logical

float / int = float

&& and | || or | ! not

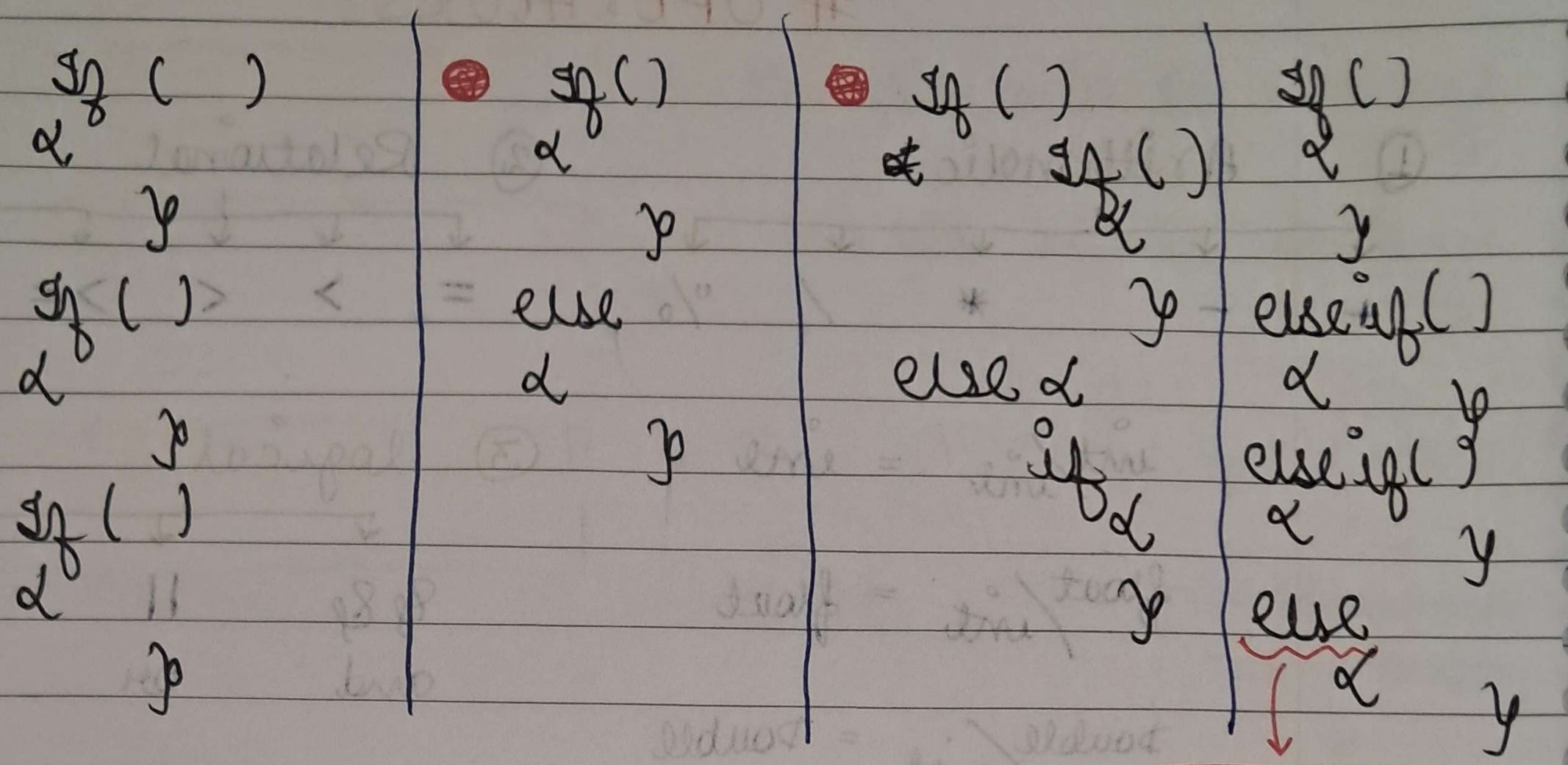
Double / int = Double

- Cin >>
  - "_" space → 32
  - "\t" tab → 10
  - "\n" enter → 9
  - → ASCII table
  - Don't Read ⊗
- ↳ a = Cin.get() ← Read ✓

Conditionals

- if (~) {
  ——
  }
  else {
  if (✓) {
  ——
  }
  else {
  ——
  }
  }

  ⇝ else if () {
  }

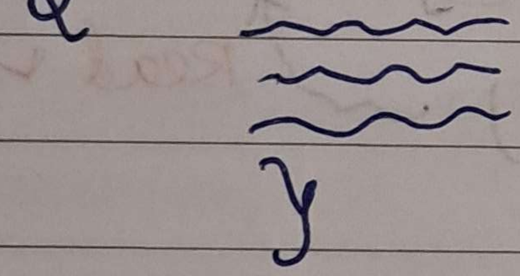


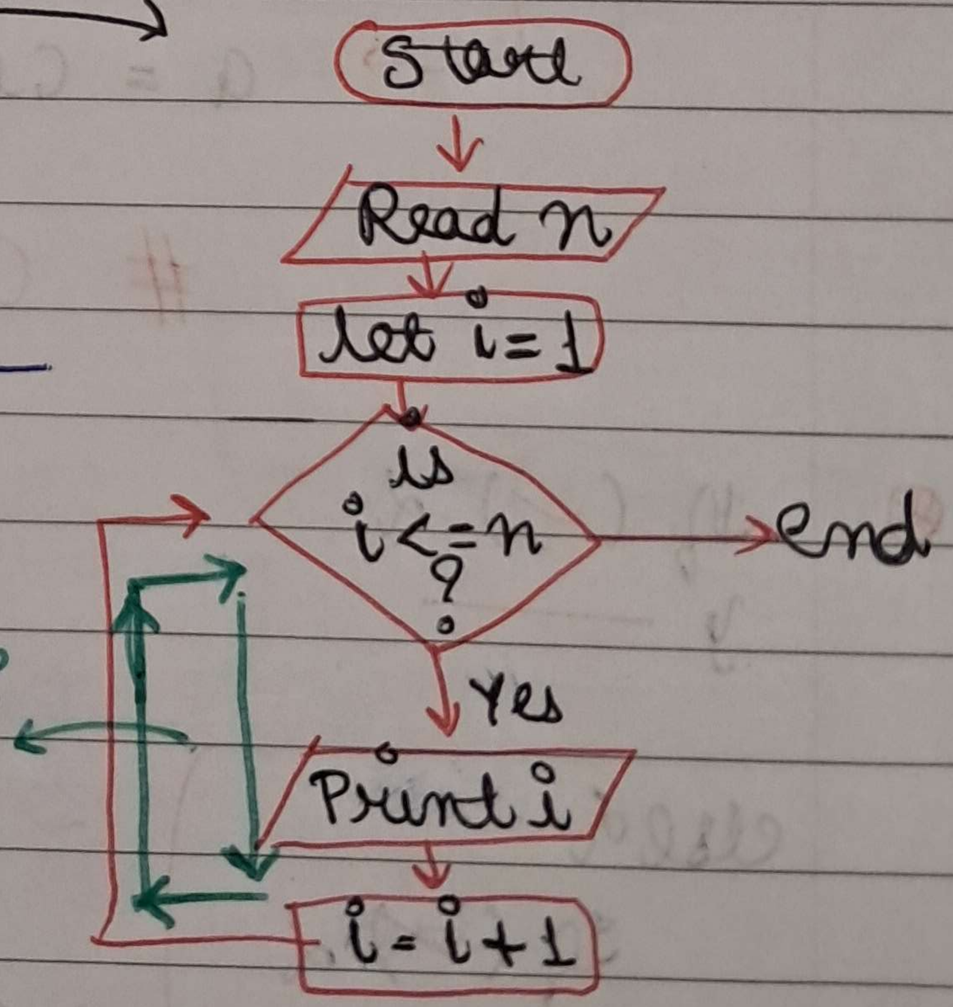
While loop.

while ( ) → जब तक यह True hai

{

}

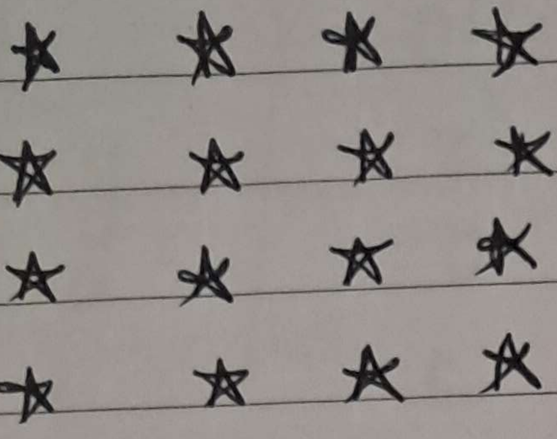





```
int n;
cin>>n;
int i=1

while(i<=n)
{  cout<<i;
   i=i+1;
}
```

Pattern Ques (Grip on loop)



Observation 1 → row = 4

Observation 2 → Har row mai kitne column Hai

↳ 1st row mai 4 colm. Hai

2nd ---- → 4

3rd ---- 4

4th ---- 4

Now find rethn B/W Row & Column

↳ Here ; → Jitne Row = Column

```
int i = 1;
while (i<=n) {
}
```

i = 1, 2, 3

O/P

```
int i = 0;
while (i<n) {
}
```

i = 0, 1, 2

LEC-4

↳ Lec 4 Have 20 (approx) pattern print problem

& Have write the approach to solve them in the code itself

↳ Using //Comment

LEC - 5

Bitwise operators

AND → &
OR → |
NOT → ~
XOR → ^

① AND → &

a = 2, b = 3

a & b → 2 → 10
3 → 11
10

o/p → 2

i/p		o/p
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

② OR → |

a = 2
b = 4

a | b
↳ 10
100
110 → o/p → 6

i/p		o/p
x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

③ NOT ~

a = 2
~a = -3

x	z
0	1
1	0

0 0 0 0 0 0 -- 0 0 1 0

~a

1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1

-3

1's comp → 0 0 0 0 0 0 0 1 0
2's comp → +1
0 0 0 0 0 0 1 1 → 3

④ XOR → ^

i/p		o/p
x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Both bits → Similar → 0
→ Dissimilar → 1

eg of XOR (^)

a = 2 → 010
b = 4 → 100

110

o/p → 6

⑤ Left Shift (<<)

5 << 1

⑤ → 00000 ---- 00101

→ 000 00 -- - 00101 0 → Padding

o/p → 10

left shift by 1 bit

3 << 2

③ → 0000 --- 0011

0000 --- 001100

o/p → 12

⑥ Right Shift (>>)

5 >> 2

⑤ → 00 00 00 -- - 00101

0 0 0000 - - - 001

Padding → o/p → 1

Right Shift
↳ Jitne Bar Shift utne Bar divide 2 se

5 >> 1 → 5/2 = 2

5 >> 2 → 5/2×2 = 5/4 = 1

- Big no → -ve
  Small → Double
- 5 × 2 → 10
  3 × 2 * 2 → 12 } Summary
- Left Shift / Right Shift
  ↳ padding with 0 (+ve)
  (-ve) no → padding
  ↳ compiler dependent.

- $i = i + 1$ → $i += 1;$
- $Sum = Sum + b$ → $Sum += b;$

__/__/____

Increment and Decrement Operator

post Inc	pre Inc	post Dec	pre Dec
$i++;$	$++i;$	$i--;$	$--i;$
Phele use kro phir inc kro	Phele implement kro then use	Phele use then decrement	int a = --1; int = 2 → i=1
int i=3, a=2; Sum = a + (i++); Sum = 2 + 3 = 5 → use then i=4	— // — Sum = a + (++i); = 3 + (3) ✓ Phele implement = 6	int a = i--; i = 3, a = 3; then i = 2	a = 1 Phele dec then store kro

for loop

```
for ( int i=0 ; i<=n ; i++ ) {
```

→ false → RUK JAYEGA

initialisation · Condn · Updating inc/dec

```
}
```

```
for ( ; ; ) {
```

Infinite loop

```
}
```

Multiple Condn check

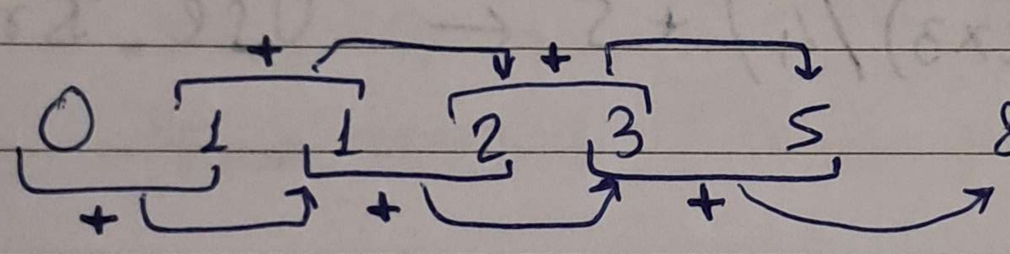
```
for ( int a=0, b=1 ; a>=0 , b>=1 ; a--, b--) {
    cout << a << " " << b;
}
```

```
int i = 1;
for (;;) {
    if (i == n) {
        cout << i << endl;
    }
    else {
        break
    }
    i++;
}
```

→ mention (x) then ∞ loop

→ get out of current loop

- fibonacci series



$$n = (n-1) + (n-2)$$

Continue → Iteration Skip

↳ continue ke neeche wala part unreachable hogaya

Scope → The lifetime of a variable, where does the variable exist and after what line of code will get destroyed.

a is accessible throughout the main after it declared

```
int main() {
    int a = 3;
    cout << a;
    if (+ve) {
        cout << a;
    }
}
```

```
int main() {
    int a = 3;
    cout << a;
    if (true) {
        int a = 5;
        cout << a;
    }
}
```

→ 'local' to int main()

→ 'local' to if block, Only accessible within the if block

```
{
  {
    b
    b
  }  Two Same variable under same scope (X)
}
```

```
{
  b
  {
    b   <- want this to be used
  }     <- Do not create this one
  b
}
```

Quickly say cout<<b;
↓
U will get upper b.

Operator Precedence → like BodMas

Brackets, we can also prioritize calculn using Bracket

① $2 \times 3/4 + 5 \longrightarrow 2 \times 0 + 5 = 5$ (int/int = int)

② $2 \times \frac{3}{4} + 5 = \frac{6}{4} + 5 = 6$ ⇒ ≠

Then $((2 \times 3)/4) + 5$ ← Use Brackets

LEC - 6

- Decimal to Binary

① Divide by 2
② Store remainder in answer
③ Repeat above 2 step until $n != 0$
④ Read Ans in Reverse order

2 → 10
5 → 101
10 → 1010
8 → 1000

- $n = 10$

Divide	Rem
$10/2 = 5$	0
$5/2 = 2$	1
$2/2 = 1$	0
$1/2 = 0$	1

Read jana ←

→ 1010