Lec-3 Let's Solve Patterns 7 Jan 2023

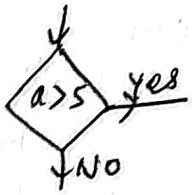
① Conditional, Loops and Patterns →

① Conditionals →

- ~~if conditionals → if statements~~
- **if condition →**

If this is true then this executes.

```
if(marks > 95){
    cout << "A grade";
}
```



Syntax :→

```
if (condition){
  // statement
}
```

```
if ( score < 300){
    cout << "India wins";
}
cout << "Pak wins";
```

let score = 325
↳ o/p → Pak wins.

score = 125
↳ o/p → India wins
Pak wins

so if 'if' condition is true then only ~~the~~ it will go to inside if condition and does the execution and move to next line.

If 'if' condition is false then we will not go inside if condition and next line will be executed.

- **if - else condition →**

```
if (age >= 18){
    cout << "You can vote";
}
else{
    cout << "You can't";
}
```

If condition is true then 'if' statement will be executed.

Otherwise 'else' statement will be executed.

- **nested if - else →**

```
if (marks >= 80){
    cout << "A";
}
else {
```

```
if (marks >= 80){
        cout << "C";
}
else {
        if (marks >= 40){
            cout << "D";
        }
        else {
            cout << "F";
        }
}
}
```

- **else if condition →**

```
if (marks >= 90){
        cout << "A";
}
else if (marks >= 80){
        cout << "B";
}
else if (marks >= 60){
        cout << "C";
}
else if (marks >= 40){
        cout << "D";
}
else {
        cout << "F";
}
```

⊙ no. of bothers

```
int bronum;
cin >> bronum;
if (bronum == 0)
    cout << "Baat Ban Jayegi";
else
    cout << "Baat Nahi Ban Payegi";
```

① **Loops** → When we want to repeat a thing multiple times then we use loops. As of now we are only studing for loops.

eg → Print your name 5 times.

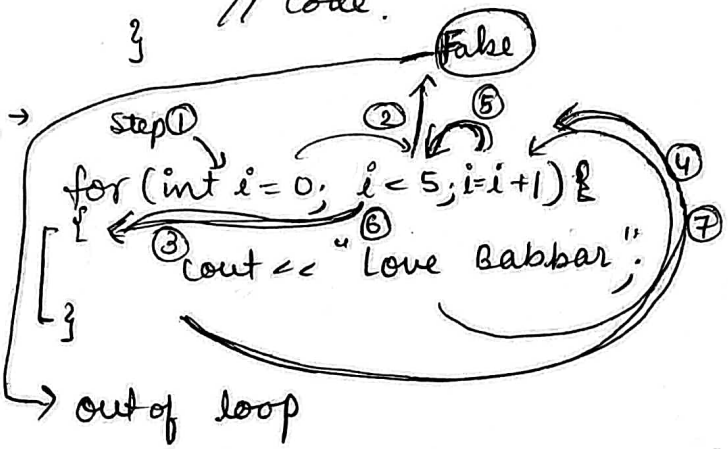
Syntax:→

```
for (int i = 0; i < 5 ; i++) {
```

(Condition: `i < 5`; Updation: `i++`; initialization: `int i = 0`)

```
// code.
}
```

flow →



$i = 0$
$0 < 5 \rightarrow T$
Love Babbar
$i = 0+1 = 1$
$2 < 5 \rightarrow T$
Love Babbar
$i = 1+1 = 2$
$2 < 5 \rightarrow T$
Love Babbar
$i = 2+1 = 3$
$3 < 5 \rightarrow T$
Love Babbar
$i = 3+1 = 4$
$4 < 5 \rightarrow T$
Love Babbar
$5 < 5 \rightarrow F$

Name will be printed.

② `for (int i = 0; i < 3; i++) {` (or `i = i+1`)

```
    cout << i;
}
```

$i = 0$	$i = 0+1 = 1$	$i = 1+1 = 2$	$i = 2+1 = 3$
$0 < 3 \rightarrow T$	$1 < 3 \rightarrow T$	$2 < 3 \rightarrow T$	$3 < 3 \rightarrow F$
print 0	print 1	print 2	out of loop.

o/p → 0 1 2

③ `for (int i = 5; i > 0; i = i-1) {`

```
    cout << i << endl;
}
```

o/p →
5
4
3
2
1

④ `for (int i = 1; i <= 10; i = i+2) {`

```
    cout << 2 * i;
}
```

o/p → 2 4 6 8 10 12 14 16 18 20

④ for (int i = 1 ; i <= 5 ; i = i+2) {

cout << i << endl;

}

o/p → 1 3 5

⑤ for (int i = 1 ; i <= 10 ; i = i*2) {

cout << i << endl;

}

i = 1	i = 1×2 = 2	i = 2×2 = 4	i = 4×2 = 8	i = 8×2 = 16
1 < 10 → T	2 < 10 → T	4 < 10 → T	8 < 10 → T	16 < 10 → False
print 1	print 2	print 4	print 8	out of loop.

⑥ for (int i = 100 ; i > 0 ; i = i/2) {

cout << i ;

}

i = 100	i = 100/2 = 50	i = 50/2 = 25	i = 25/2 = 12	i = 12/2 = 6	i = 6/2 = 3	i = 3/2 = 1
100 > 0 → T	50 > 0 → T	25 > 0 → T	12 > 0 → T	6 > 0 → T	3 > 0 → T	1 > 0 → T
print 100	print 50	print 25	print 12	print 6	print 3	print 1

i = 1/2 = 0

0 > 0 → F

out of loop.

o/p → 100 50 25 12 6 3 1

⑦ for (int i = 5; (i >= 0 && i <= 10); i = i+1) {

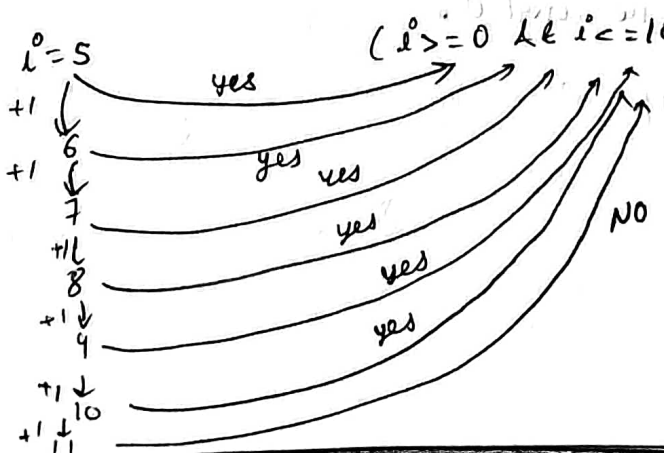
cout << i;

}

i = 5	i = 6
5 >= 0 && 5 <= 10 → T	6 >= 0 && 6 <= 10 → T
print 5	print 6

, similarly for 7, 8, 9, 10

o/p → 5 6 7 8 9 10



⊙ Let see what is mandatory in for loop's syntax →

```
for (int ①i=0; ②i<5; ③i=i+1){
        cout<<i;          //o/p → 0, 1, 2, 3, 4
}
```

①
```
int i=0;
for( ; i<5; i=i+1){
    cout<<i;          //o/p → 0 1 2 3 4
}
```

②
```
int i=0;
for( ; ; i=i+1){
    if(i<5){
        cout<<i;      //o/p → 0 1 2 3 4
    }
}
```

③
```
int i=0;
for( ; ; ){
    if(i<5){
        cout<<i;      //o/p → 0 1 2 3 4
    }
    i=i+1;
}
```

So nothing is mandatory only semicolons are.

H.W

① what happens →

```
• int n;
  if(cin>>n){
      cout<<"Babbar";
  }
```

&

```
• int n;
  if(cout<<"Babbar"){
      cout<<"Love";
  }
```

Check for all +ve, -ve and 0.

⊙ Patterns →

• Solid Rectangle →

Observation → Steps

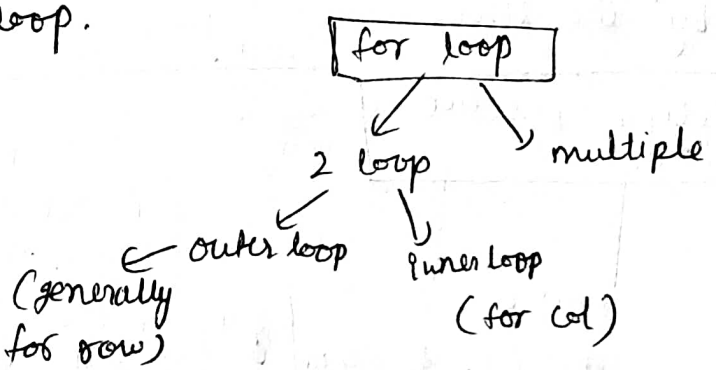
① row - observation
total rows = 3

② col - observation
col 0 → 3 ✯
col 1 = 3 ✯
col 2 = 3 ✯
col 3 = 3 ✯
col 4 = 5 ✯

in each column we have to print 3 ✯.

or a diff observation is that we will print 5 ✯ in each row.

we print patterns by for loops. Generally 2 nested loop.




```
// outer loop
for (int row = 0; row < 3 ; row++){
    //inner loop → for each column 3 ✯.
    for (int col = 0; col < 5; col = col+1){
        cout << "✯";
    }
    cout << endl;  // Because we are going to next line after each row.
}
```

dry run →

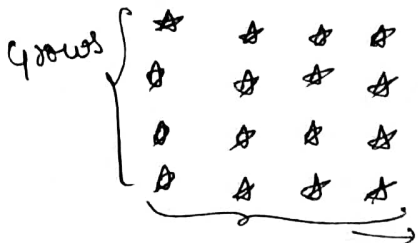
```
row = 0
0 < 3 → T
  col = 0
  0 < 5 → T
  col = 1
  1 < 5 → T
  col = 2
  2 < 5 → T
  col = 3
  3 < 5 → T
  col = 4
  4 < 5 → T
  col = 5
  5 < 5 → F
row = 1
1 < 3 → T
  col = 0
  0 < 5 → T
  col = 1
  1 < 5 → T
  col = 2
  2 < 5 → T
  col = 3
  3 < 5 → T
  col = 4
  4 < 5 → T
  col = 5
  5 < 5 → F
row = 2
2 < 3 → T
  col = 0
  0 < 5 → T
  col = 1
  1 < 5 → T
  col = 2
  2 < 5 → T
  col = 3
  3 < 5 → T
  col = 4
  4 < 5 → T
  col = 5
  5 < 5 → F
row = 3
3 < 3 → F → out of loop.
```

T → print ✯
F → break the loop

o/p →

✯ ✯ ✯ ✯ ✯
✯ ✯ ✯ ✯ ✯
✯ ✯ ✯ ✯ ✯

• Square Pattern →



→ 4 row

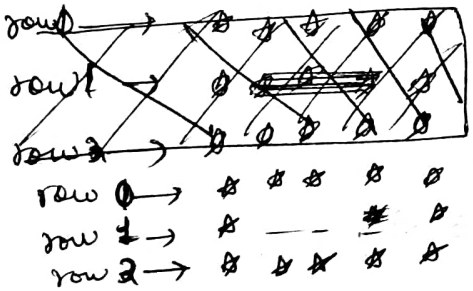
→ row 0 → 4 ☆
row 1 → 4 ☆
row 2 → 4 ☆
row 3 → 4 ☆
and in each loop 4 ☆ are printed.

```
for (int row = 0; row < 4; row++){
    for (int col = 0; col < 4; col++){
        cout << " ☆ ";
    }
    cout << endl;
}
```

We can make this genric. by using n instead of 4, n is input taken by the user.

using goto → ghatiya practice (bad)

⊙ Hollow Rectangle



row 0 → 5 ☆
row 1 → 2 ☆, 3 space
row 2 → 5 ☆
} 1st and last row → 5 ☆ (all)

In remaining rows only two ☆ (one at the start, one at the end other are stars).

In each row 5 things are printed →

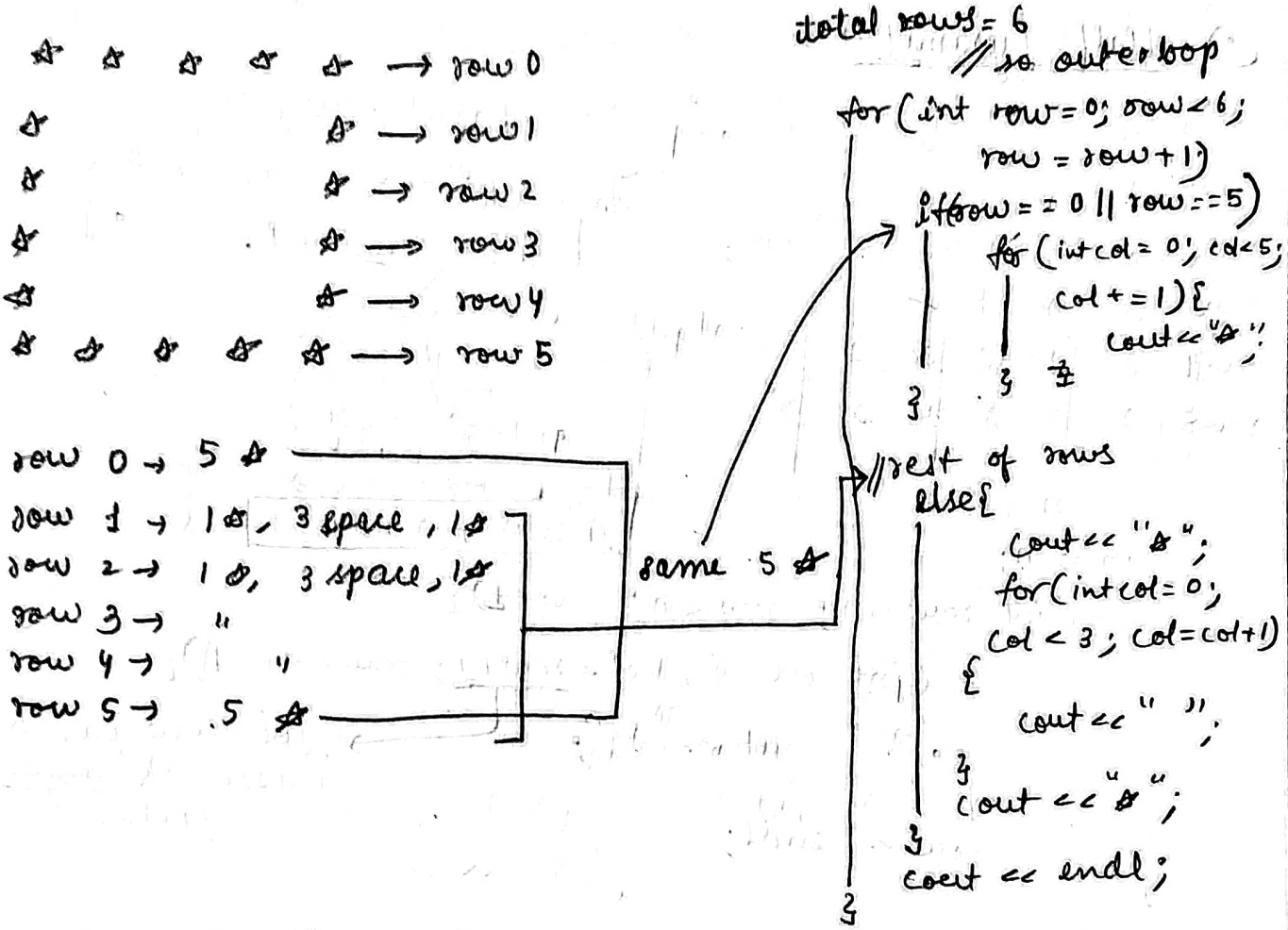
row 0 → 5 ☆
row 1 → 2 ☆, 3 space
row 2 → 5 ☆

code -

```
for (int row = 0; row < 3; row++){        // no. of rows
    if (row == 0 || row == 2){            // 1st and last row
        for (int col = 0; col < 5; col++){
            cout << " ☆ ";                // no. of column
        }
        cout << endl;
    }
    else {                                // remaining rows.
```

```
        cout << "*";          // 1st star
        for (int i = 0; i < 3; i = i+1){   // middle spaces.
            cout << " ";
        }                     // no. of col-2
        cout << "*";          // last star
    }
    cout << endl;
}
```

(1)



(2) Let's make it generic →

We will make two variables and store no. of rows and no. of columns in ~~that~~ these variables.

```
int rowCount, colCount;
cin >> rowCount >> colCount;
for (int row = 0; row < rowCount; row += 1){
    if (row == 0 || row == rowCount - 1){
        for (int col = 0; col < colCount; col += 1){
            cout << "*";
        }
    }
}
```

```
else {
        cout << "*";
        for (int col = 0; col < colcount-2; col+=1) {
                cout << " ";
        }
        cout << "*";
    }
    cout << endl;
}
```

⊙ Half Pyramid →

row→0 *
row→1 * *
row 2 * * *
row 3 * * * *
row 4 * * * * *
row 5 * * * * * *

total row = n = 6

In each row stars printed are rowno + 1.

row 0 → 1 *
row 1 → 2 *
row 2 → 3 *
row 3 → 4 *
row 4 → 5 *
row 5 → 6 *

n^{th} row → n+1 *

```
int n;
cin >> n;
for (int row = 0; row < n; row+=1) {
        for (int col = 0; col < row+1; col+=1) {
                cout << "*";
        }
        cout << endl;
}
```

→ jis row par hai usse ek ~~jayda~~ jyada.

⊙ Inverted Half Pyramid →

row 0→ * * * * * *
row 1→ * * * * *
row 2→ * * * *
row 3→ * * *
row 4→ * *
row 5→ *

row 0 → 6 * = $n - row$ (6 − 0)
row 1 → 5 * = $n - row$ (6 − 1)
row 2 → 4 * = $n - row$ (6 − 2)
row 3 → 3 * = $n - row$ (6 − 3)
row 4 → 2 * = $n - row$ (6 − 4)
row 5 → 1 * = $n - row$ (6 − 5)

In each row (n − row) stars.

```
int n;
cin >> n;
for (int row = 0; row < n; row = row+1) {
```

```
for (int col = 0; col < n - row; col = col + 1){
        cout << "*";
    }
    cout << endl;
}
```

Step ① row count → row loop
Step ② row logic break - find formula. } Two main steps.
Step ③ - -

⊙ Numeric Half Pyramid →

row 0 → 1
row 1 → 1 2
row 2 → 1 2 3
row 3 → 1 2 3 4
row 4 → 1 2 3 4 5
col 0 col 1 col 2 col 3 col 4

row 0 → count till 0+1=1
row 1 → count till 1+1=2
row 2 → count till 2+1=3
row 3 → count till 3+1=4
row 4 → count till 4+1=5
} row +1

col 0 → 1
col 1 → 2
col 2 → 3
col 3 → 4
col 4 → 5
} col no. + 1

```
int n;
cin>>n;
for(int row=0; row<n; row++){
    for(int col=0; col<row+1; col++){
        cout << col+1;
    }
    cout << endl;
}
```

Dry run →

n=3
row = 0
0 < 3 → T
col = 0
0 < 0+1 → T
↳ print col+1 = 1
col = 1
1 < 1 → F
row = 1
1 < 3 → T
col = 0
0 < 1+1 → T
↳ print (col+1 = 0+1 = 1
col = 1
1 < 2 → T ↳ print col+1 = 1+1 = 2
col = 2
2 < 2 → F
row = 2
2 < 3 → T
col = 0
0 < 3 → T → print 1
col = 1
1 < 3 → T → print 2
col = 2
2 < 3 → T → print 3
col = 3
3 < 3 → F → out of inner loop
row = 3, 3 < 3 → F → out of outer loop.

```
1
1 2
1 2 3
```

⊙ Inverted Half Numeric Pyramid :-

```
1 2 3 4 5      n=5
1 2 3 4
1 2 3
1 2
1
```

total row = n = 5

row 0 → 1, 2, 3, 4, 5
row 1 → 1, 2, 3, 4
row 2 → 1, 2, 3
row 3 → 1, 2
row 4 → 1

} n-row

and each col, col+1 is printed.

```
int n;
cin>>n;
for(int row=0; row<n ; row+=1){
        for(int col=0; col<n-row; col+=1){
                cout<<col+1;
        }
        cout<<endl;
}
```

Try Full Pyramid and ~~Fadt~~ Inverted Full Pyramid.