29-01-2023

Date.....

Patterns → increase logic building
→ strong the concept of loop!.

(1) Print solid Rectangle

```
* * * * *
* * * * *
* * * * *
* * * * *
```

Think process of patterns,
Step 1, observe the Rows, i.e., How many Rows are in given pattern.

```
* * * * *   - row 0
* * * * *   - row 1
* * * * *   - row 2
* * * * *   - row 3
```

} Total 4 Rows

Step 2, column observation, i.e., How many columns are in given pattern.

```
* * * * *
* * * * *
* * * * *
* * * * *
```

col.0 col.1 col.2 col.3 col.4

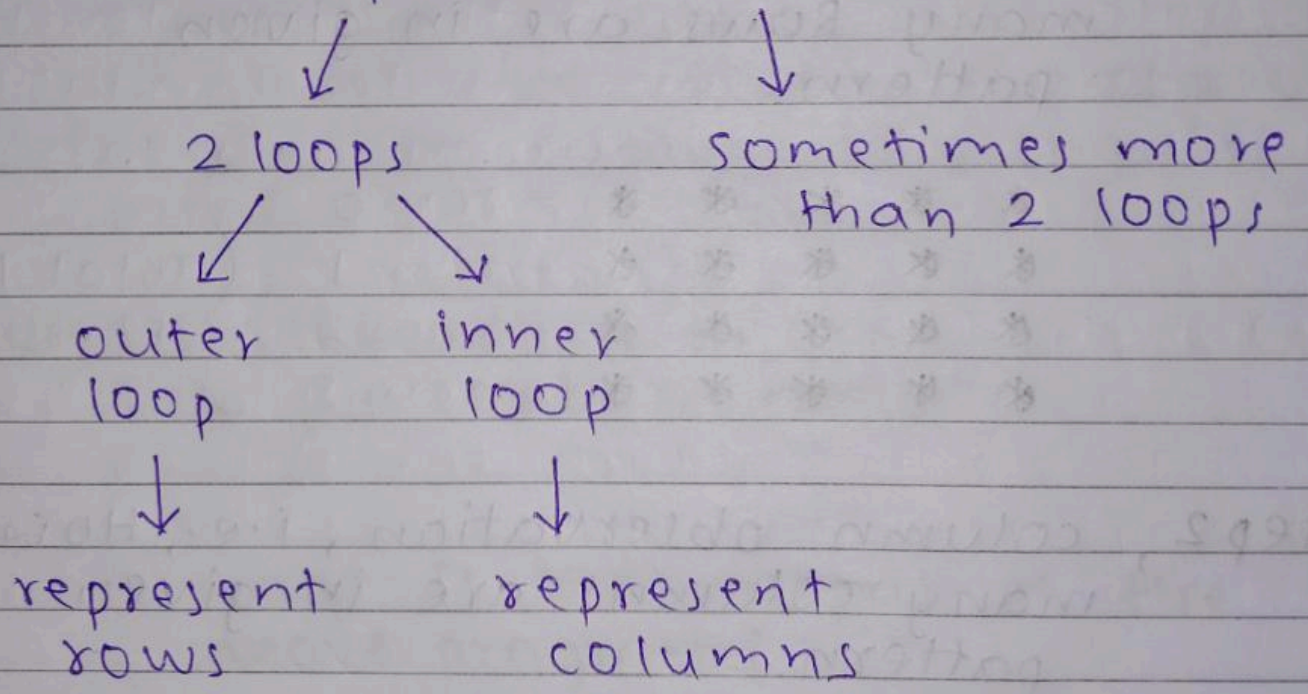
Total 5 columns

Date.....

Step3, now check how many stars (*) are printed in each column.

column 0 - 3 stars (*)
column 1 - 3 stars (*)
column 2 - 3 stars (*)
column 3 - 3 stars (*)
column 4 - 3 stars (*)

Generally, patterns is printed using Nested Loops



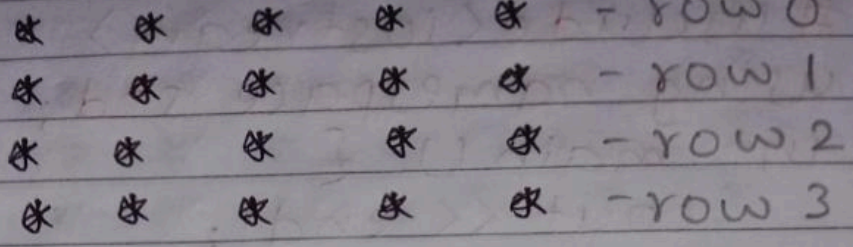
Eg:-

```
for () // outer loop for rows
{
    for () // inner loop for columns
    {

    }
}
```

Date.....

① Print solid Rectangle



step1, First create the outer loop for representing rows, so

```
for(int i = 0; i < 4; i++){

}
```

↓ this loop is run for i = 0, then i = 1, then i = 2 & laitly i = 3. Means, total 4 times for 4 rows.

Now, in each row we have 5 stars (*) means we have to create a loop that will run 5 times for each row.

step 2, create inner loop which will run 5 times for each row. So,

```
for(int j = 0; j < 5; j++){
    cout << " * ";
}
```

Date.....

code:-

```
#include<iostream>
using namespace std;
int main() {
    cout << endl;
    for(int row = 0; row < 4;
                        row++)
    {
        for(int col = 0; col < 5;
                            col++)
        {
            cout << "*   ";
        }
        cout << endl;
    }
    return 0;
}
```

Date.....

(2) Print solid square.

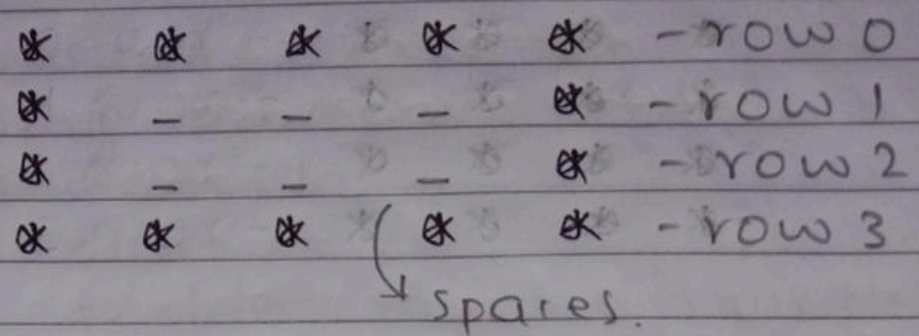
```
* * * *  - row 0
* * * *  - row 1
* * * *  - row 2
* * * *  - row 3
```

code:-

```
#include <iostream>
using namespace std;
int main() {
    cout << endl;
    for(int r = 0; r < 4; r++){
        for(int c = 0; c < 4; c++){
            cout << " * ";
        }
        cout << endl;
    }
    return 0;
}
```

Date.....

(3) Hollow Rectangle.



Step 1, check how many stars (*) are printed in each row.

row 0 - 5 stars (*)
row 1 - 1st place me star (*)
2nd place me space ( )
3rd place me space ( )
4th place me space ( )
5th place me star (*)
row 2 - 1st place me star (*)
2nd place me space ( )
3rd place me space ( )
4th place me space ( )
5th place me star (*)
row 3 - 5 stars (*)

There are 5 things printed in each row, i.e.,
row 0 - 5 stars (*)
row 1 - 1 star (*), 3 spaces ( ), 1 star
row 2 - 1 star (*), 3 spaces ( ), 1 star
row 3 - 5 stars (*)

Date.....

Outer loop,

```
for( int i = 0; i < 4; i++) {
}
```

Inner loop,

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

Condition to print row 1 & row 2, i.e; print spaces (_)

```
if( row == 0 || row == 3) {
    print 5 stars (*)
}
else {
    print 1 star (*)
    print 3 spaces (_)
    print 1 star (*)
}
```

Date.....

code :-

```
#include<iostream>
using namespace std;
int main(){
    cout << endl;
    for(int r=0; r<4; r++){
        if(r==0 || r==3){
            for(int c=0; c<5; c++)
            {
                cout<<"*    ";
            }
        }
        else{
            cout<<"*    ";
            for(int c=0; c<3; c++)
            {
                cout<<"    ";
            }
            cout<<"*    ";
        }
        cout<<endl;
    }
    return 0;
}
```

Date.....

Generic code :-

```
#include<iostream>
using namespace std;
int main(){
    cout<<endl;
    int rcount, ccount;
    cout<<"Enter rows = ";
    cin>>rcount;
    cout<<"Enter column = ";
    cin>>ccount;
    for(int r=0; r<rcount;
                    r++)
    {
        if(r==0 || r==rcount-1)
        {
            for(int c=0; c<ccount;
                        c++)
            {
                cout<<"* ";
            }
        }
        else{
            cout<<"* ";
            for(int c=0; c<ccount
                    -2; c++)
            {
                cout<<" ";
            }
            cout<<"* ";
        }
        cout<<endl;
    }
}
```


Teacher's Sign .....

Date.....

(4) Half Pyramid.

```
*                  - row 0
* *                - row 1
* * *              - row 2
* * * *            - row 3
* * * * *          - row 4
* * * * * *        - row 5
```

Step 1, check how many stars (*) are printed in each row.

row 0 - 1 star (*)
row 1 - 2 star (*)
row 2 - 3 star (*)
row 3 - 4 star (*)
row 4 - 5 star (*)
row 5 - 6 star (*)

How many stars are printed in each row? (Formula)
→ Stars that are printed in each row are row no. + 1, i.e.,

row 0 = row 0 + 1 = 1 (*)
row 1 = row 1 + 1 = 2 (*)
row 2 = row 2 + 1 = 3 (*)
row 3 = row 3 + 1 = 4 (*)
row 4 = row 4 + 1 = 5 (*)
row 5 = row 5 + 1 = 6 (*)

Date.....

code :-

```
#include <iostream>
using namespace std;
int main()
{
    int rcount;
    cout << "\n enter rows = ";
    cin >> rcount;
    for(int r = 0; r < rcount; r++)
    {
        for(int c = 0; c < r+1; c++)
        {
            cout << "* ";
        }
        cout << endl;
    }
    return 0;
}
```

Date.....

(5) Inverted Half pyramid.

```
* * * * * *   - row 0
* * * * *     - row 1
* * * *       - row 2
* * *         - row 3
* *           - row 4
*             - row 5
```

Step1, check how many stars (*) are printed in each row,

row 0 - 6 stars (*)
row 1 - 5 stars (*)
row 2 - 4 stars (*)
row 3 - 3 stars (*)
row 4 - 2 stars (*)
row 5 - 1 star (*)

How many stars are printed in each row? (Formula)

→ stars that are printed in each row are total rows - row no.

row 0 = 6 - 0, i.e., 6 stars (*)
row 1 = 6 - 1, i.e., 5 stars (*)
row 2 = 6 - 2, i.e., 4 stars (*)
row 3 = 6 - 3, i.e., 3 stars (*)
row 4 = 6 - 4, i.e., 2 stars (*)
row 5 = 6 - 5, i.e., 1 star (*)

Date.....

code :-

```
#include <iostream>
using namespace std;
int main()
{
    int rcount;
    cout << "\n enter rows = ";
    cin >> rcount;
    for(int r = 0; r < rcount;
                        r++)
    {
        for(int c = 0; c < rcount - r;
                            c++)
        {
            cout << " * ";
        }
        cout << endl;
    }
    return 0;
}
```

Date.....

(6) Numeric Half Pyramid.

```
1              - row 0
1 2            - row 1
1 2 3          - row 2
1 2 3 4        - row 3
1 2 3 4 5      - row 4
```

Step 1, For Half pyramid case we always use "row no. + 1" formula, i.e., inner loop will always run from 0 to row no. + 1.

Step 2, In each column, we're printing "column no. + 1" st number, i.e.,

col. 0 : 0 + 1 : 1 print
col. 1 : 1 + 1 : 2 print
col. 2 : 2 + 1 : 3 print
col. 3 : 3 + 1 : 4 print
col. 4 : 4 + 1 : 5 print.

Date.....

code!-

```
#include <iostream>
using namespace std;
int main()
{
   int rcount;
   cout << "\n enter rows = ";
   cin >> rcount;
   for(int r = 0; r < rcount; r++)
   {
      for(int c = 0; c < r+1; c++)
      {
         cout << c+1;
      }
      cout << endl;
   }
   return 0;
}
```

Date.....

(7) Inverted numeric Half Pyramid.

```
1 2 3 4 5   - row 0
1 2 3 4     - row 1
1 2 3       - row 2
1 2         - row 3
1           - row 4
```

Step 1, for inverted case, we always use "total rows - row no." formulae, i.e., inner loop will always run from 0 to total row - row no.

Step 2, In each column, we are printing "column no. + 1" st number, i.e.,

col. 0 : 0 + 1 = 1 print
col. 1 : 1 + 1 = 2 print
col. 2 : 2 + 1 = 3 print
col. 3 : 3 + 1 = 4 print
col. 4 : 4 + 1 = 5 print

Date.....

code:-

```
#include<iostream>
using namespace std;
int main()
{
    int rcount;
    cout<<"enter row :";
    cin>>rcount;
    for(int r=0; r<rcount;
                    r++)
    {
        for(int c=0; c<rcount-
                r; c++)
        {
            cout<<c+1;
        }
        cout<<endl;
    }
    return 0;
}
```