L6 - Pattern Printing

→ Hollow Pyramid Patteen

```
n=1     n=2       n=3        n=4
                               *
 *       *         *          * - *
       * * *      * - *      * - - - *
                * * * * *   * * * * * *
```

```
n=5     - - - - - *              I
         - - - * - *             II
         - - * - - - *           III
         - * - - - - - *         IV
         * * * * * * * * *       V
```

① n=5

② No of rows=5

③ No of cols= (vary)

④ Analysis

n=5 r=1 - I^st Row - $\overset{n-row}{4space} + 1*$ — r==1

n=5 r=2 - IInd Row = 3 spaces + 1* + $\overset{2(r)-3}{1sp}$ + 1*

n=5 r=3 - III^rd Row = 2sp + 1* + 3sp + 1*

n=5 r=4 - IV^th Row = 1sp + 1* + 5sp + 1*

n=5 r=5 - V^th Row = 0sp + 9 staa.

≃ r==n

(2(n)-1) staes

②

n=4

```
_ _ _ *          - I
_ _ * _ *        - II
_ * _ _ _ *      - III
* * * * * * *    - IV
```

① n=4 r=1 Ist Row — 3spaces + 1 * ~~[illegible]~~ 2(r)-3

⇑

② n=4 r=2 IInd Row - 2spaces + 1* + 1sp + 1*

③ n=4 r=3 IIIrd Row - 1space + 1* + 3sp + 1*

④ n=4 r=4 IVth Row - 0space + 7*

⇓ ⇓

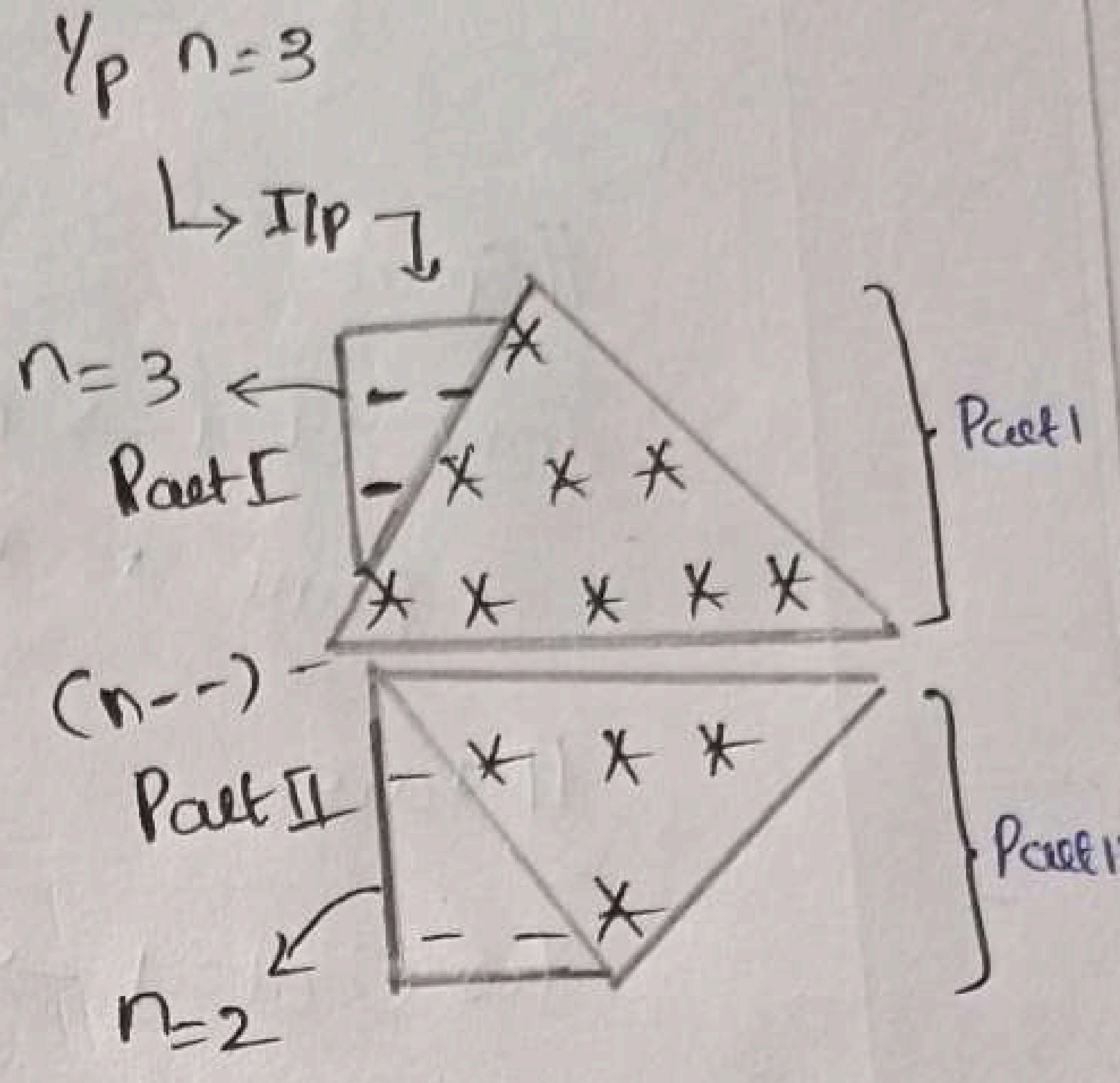
n-row   if r==n

2(n)-1

②

Solid Diamond Pattern

Ⓘ Part I



① n=3

② No of rows = 3

③ No of cols = (vary)

④ Analysis

n=3 r=1 - Ist Row - 2sp+1*

n=3 r=2 - IInd Row - 1sp+3*

n=3 r=2 - IIIrd Row - 0sp+5*

(n-row) (2(row)-1)

- Decrement n by 1 because below there are two rows

n--

Ⓘⓘ Part II

① n=2

_ * * *

- - *

② No of rows = 2

③ No of cols = (vary)

④ Analysis

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

n=2 r=1 - Ist Row - 1sp + 3*

n=2 r=2 - IInd Row - 2sp + 1*

I

```
for (row 1 →  <=n)
{
    //space
    for ( col = 1 → n-row)
        print " "
    // stars
    for (col = 1 → 2(row)-1)
        print "*"
```

n- -

II

```
for ( row 1 → <= n )
{
    //space
    for ( col = 1 → col<= row)
        print " "
    //stars
    for ( col = 1 → col<=(2(n-r)+1)
        print "*"
```

⑤
method-2

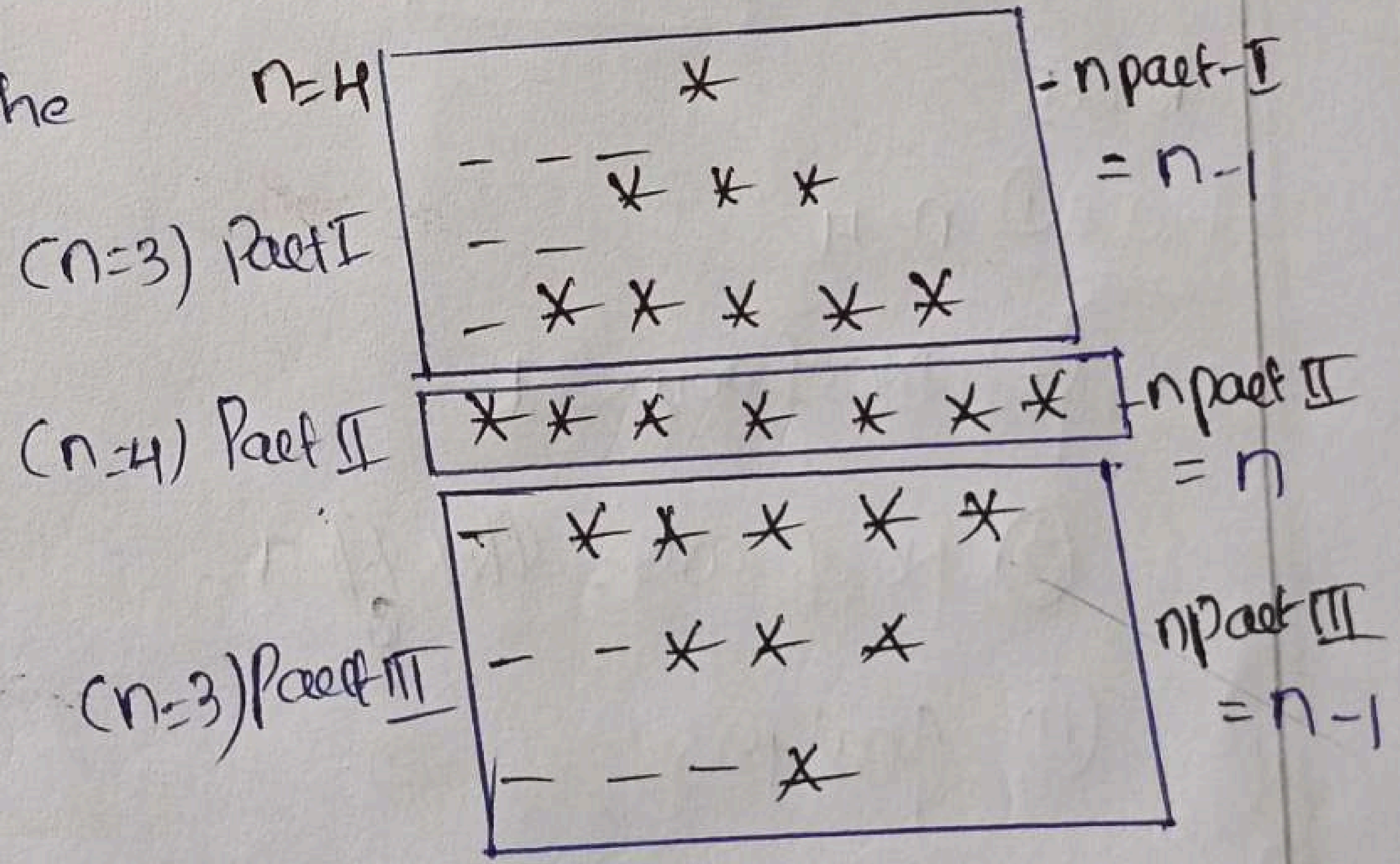
- lets say if unable to findout the formula of stars in part II then take one count variable

```
                                              use this
        int totalStars= 2*n-3  -> outside.for loop of row

      for(int col=1; col<= totalStars; col++)
      {
          print "*"
            ↓
          cout<<

      }
          totalStars = totalStars-2;
```

Stars -> ko jab tak totalStars 1 nhi ho jata tab tak print karte rehna hai

method-3

* whenever we divide the



Part I

① n=3

② No of rows = 3

③ No of cols = (vary)

n=3 r=1 Ist row - 3sp+1*

n=3 r=2 IInd row - 2sp+3*

n=3 r=3 IIIrd row - 1sp+5*

(2(1)-1)

⑥

npartI — ⓝ=3

```
for(int r=1; r<=n; r++)
{
        //spaces
   for( int col=1; col<=(n-r+1); col++)
   {
         cout<<" ";
   }
        //Stars
    for( int col=1; col<=2*(r)-1; col++)
    {
          cout<<"*";
     }
}
```

PartII

① n=4

② Noof rows=1

③ No of cols= [illegible] 7

④ Analysis

n=4 r=1 - Ist row → ~~7~~* — 2*n-1

```
for (col=1; col<=2(n)-1; col++)
{
      cout<<"*";
}
coutzendl;
```

Paet III

n=3

① n=3

② No of rows = 3

③ No of cols = (vary)

④ Analysis

n=3	r=1	Ist Row	— 1space + 5*
n=3	r=2	IInd Row	— 2space + 3*
n=3	r=3	IIIrd Row	— 3space + 1*

3space ↓ $\boxed{(=r)}$

1* ↓ oddnumbee (2x-1, 2(x)+1) ↓ $\boxed{2(n-r)+1}$

(n-r)
2(x) +1

5

3

1

```
for (int row=1; row<=n; row++)
{
    for (int col=1; col <= ~~[illegible]~~ r +1; col++)
    {
        cout << " ";
    }
    for (int col=1; col <= 2(n-r) +1; col++)
    {
        cout << "*";
    }
}
```

(3) Hollow Diamond Pattern

I/P n = 3

↳ O/P

(n = 3) Part I

```
_ _ *
_ * _ *
```

npart I = n-1

(n = 3) Part II

```
* _ _ _ *
```

npart II = n

(n = 2) Part III

```
_ * _ *
_ _ *
```

npart III = n-1

Part I

(1) npart I = n-1 = 2

(2) No of rows = 2

(3) No of cols = vary

(4) Analysis

n = 2 r = 1, I^st Row − 2sp + 1* (r==1)

n = 2 r = 2 II^nd Row − 1sp + 1* + 1sp + 1*

$\boxed{n-r+1}$ *   2(r-1)-1

(2(r-n)+1)?

r=2	1
r=3	3
r=4	5

2(r-1) - 1 ✓

$\boxed{2(r)-3}$

Part II

(1) npart II = n = 3

(2) No of rows = 1

(3) Analysis

n = 3 − r = 1 − 1^st Row ⇒ (1)* + 3sp + (1)*

↓

2(r-1) 1 or 2(n-1)-1

?

Part III

① n=2

② No of rows = 2

③ No of cols = (vary)

④ Analysis

n=2 r=1 – Ist Row – 1sp + 1* + 1sp + 1*

n=2 r=2 – IInd Row – 2sp + $\boxed{1*}$

$\underset{\boxed{\text{row}}}{\overset{}{2\text{sp}}}$ = row, $\boxed{1*}$ = $\boxed{r==n}$

→ Method – II

Part I

① n=4

② No of rows = 4

③ No of cols = (vary)

④ Analysis

```
n=4         ___*            Part I
            __*_*
nPart I     _*___*
=n          *_____*
=4
            _*___*          Part II
nPart II    __*_*
=n-1        ___*
=3
```

			Spaces (r==1)	
r=1	Ist	–	3sp + I*	
r=2	IInd	–	2sp + 1* + $\boxed{1\text{sp}}$ + 1*	middle rows
r=3	IIIrd	–	1sp + 1* + $\boxed{3\text{sp}}$ + 1*	middle rows
r=4	IVth	–	0 spaces + 1* + $\boxed{5\text{sp}}$ + 1*	middle rows

npart I = $\boxed{4}$

0 spaces = (n-r)

middle rows: 1* Spaces ⇒ 2(r-1)-1 1*

2x-1, 2x+1

spaces

1 r=2
3 r=3
5 r=4

} 2(row-1)-1

```
    int part1 = n;

for(int row=1; row<=npartI; row++)
{
    //spaces
    for(int col=1; col<=(npartI-row);col++)
    {
        cout<<" ";
    }
    // star
    if(row==1)
    {
        cout<<"*";
    }
    else
    {
        //stars
        cout<<"*";
        //spaces
        for(int col=1; col<=(2*(row-1)-1);col++)
        {
            cout<<" ";
        }
//stars
~~for~~ cout<<"*";
    }
    cout<<endl;
  }
```

Part II

- no of spaces at row=4 we have to find out (because It is last row).

- if we get the last row (in part1) spaces ~~(in part)~~ we can easily get next row (which is at part 2) Spaces.

How? (Subract '2' from last row spaces)

Let int maxSpace = $(2*(row)-3)$

$= 2(4)-3$

$= 8-3 = \boxed{5}$ spaces at row 4

row1 = maxSpace - 2 ⇒ (no of spaces = 3) of part II

① n=3 (npart II = 3)

② No of rows = 3

③ No of cols = (vary)

~~{(if(r<=n))]~~

npaet II = 3, r=1 I - 1sp + 1* + $\underline{\underline{3}}$sp + 1* — - * - - - *

npaet II = 3, r= 2 II - 2sp + 1* + $\underline{1}$sp + 1* — - - * - *

- - - *

npaet II = 3, r= 3 $\underline{\text{III}}$ - 3sp + 1*

↳ if(r==n)

cout<<"*"

=r

<u>First</u> - 1*

<u>Spaces</u>

↳ $\boxed{2(n-r)-1}$ or we can use $\boxed{\text{maxSpace}-2}$

<u>First</u> - 1*

```
int maxSpace = 2*n - 3

int nPaet2 = n-1

for(int row=1; row<=nPaet2; row++)

{
   //Spaces
   for(int col=1; col<=row; col++)
   {
        cout<<" ";
   }
   // Staes  if(row==1)
              {
               cout<<"*";
              }
```

```
else
{
   cout<<"*";

   //Spaces___ maxSpace= maxSpace-2
   for(int col=1; col<=maxSpace; col++)
         cout<<" ";

      cout<<"*";
}
```

④ Rhombus Pattern

I/p → n=4

```
      * * * *
_ _ _
    * * * *
_ _
  * * * *
_
* * * *
```

① n=4

② No of rows=4

③ No of cols= (vary)

④ Analysis

			Space		Star
n=4	row=1	Ist Row -	3sp	+	4*
n=4	row=2	IInd Row -	2sp	+	4*
n=4	row=3	IIIrd Row -	1sp	+	4*
n=4	row=4	IVth Row -	0sp	+	4*
			↓ (n-r)		↓ (n)

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

(5)

→ Hour-Glass Pattern.

I/p - n=3

L O/p -

Part I

```
* * * * * *
- * * * *
- - * *
```

Part II

```
- - * *
- * * * *
* * * * * *
```

Part I

① n=3

② No of rows = 3

③ No of cols = (vary)

n=3 - r=1 - Ist Row - 0sp + 6*

n=3 r=2 - IInd Row - 1sp + 4*

n=3 r=3 - IIIrd Row - 2sp + 2*

(r-1) ↓ 2(x) = 2*(n-r+1)

$2*(x)$ 6

$\downarrow$ 4

$2*(n-r+1)$ 2

Part II

① n=3

② No of rows = 3

③ No of cols = (vary)

④ Analysis

n=3 r=1 I^{st} Row - 2sp + 2*

n=3 r=2 II^{nd} Row - 1sp + 4*

n=3 r=3 III^{rd} Row - 0sp + 6*

$\Downarrow$ $\Downarrow$

$(n-r)$ $2(x)$

$\downarrow$

r

⑥

→ Zig-Zag Patteen

I/p → n=3
↳ I/p
↳

Paet I

nPaetI = n = ③

① n=3

② No of rows = 3

③ No of columns = (vaey) = n = ③   n PaetI

④ Anaeysis   spaces

n=3 r=1 Ist - 0sp+
n=3 r=2 IInd - 1sp+
n=3 r=3 IIIrd - 2sp+
↓
(r-1)

```
* - - - - - - *
- * - - - - *
- - * - *
- - - *
- - * - *
- * - - - - *
* - - - - - - *
```

n Paet II = n+1 = ④

Staest spaces
— *
— spaces — 2(n-r)+1
(or)
take
— *

take maxspace = 2(n)-1
& decrement by 2

5
3
1

```
int  nPartI = n;
int maxspace = 2*n-1;

for(int row=1; row<=nPartI;row++)
{
    //spaces
    for(int col=1; col<=(row-1) ; col++)
          cout<< "  " ;

     //stars + spaces

     //stars
     cout<<"*";
          //spaces
     for(int col=1; col<=2(n-r)+1; col++)
        {
            cout<< "  " ;
        }
            cout<<"*";

            cout<<endl;

}
```

Paet II

npaetII = 4

```
---*
--*-*
-*---*
*-----*
```

① n = 4

② Noof rows = 4

③ Noof cols = (vaey)

④ Analysis

$$\overset{(n-row)}{3sp} + 1* \quad (r==1)$$

nPaetII = 4 - row = 1 - Ist Row - 3sp + 1* (r==1)

nPaetII = 4 - row = 2 - IInd Row - 2sp + 1* + 1sp + 1*

nPaetIII = 4 - row = 3 - IIIrd Row - 1sp + 1* + 3sp + 1*

nPaet IV = 4 - row = 4 - IVth Row - 0sp + 1* + 5sp + 1*

```
int nPaetII = n+1;
for(int row=1; row<=nPaetII; row++)
{
    // spaces - for(col=1; col<=(n-row);
                        col++)
                { " " }
    // staes
        if (row==1) print "*"
        else {
            cout << "*"
            for ( col=1; col<=(2(r-2)+1) ; col++)
                cout << " "
            cout << "*"; }
} cout << endl;
```

- *
- spaces - 2(r-2)+1 → mine
  or
  2(r-1) - 1
  ↳ (love sir's)
  or
  you can take maxspace.
- *