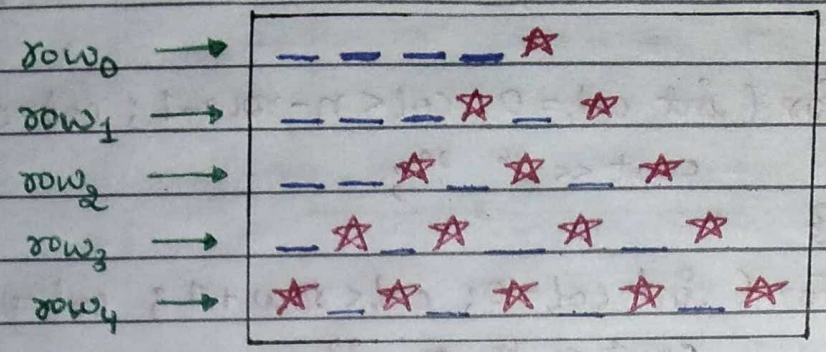
BASICS OF PROGRAMMING

LEVEL-2 [PATTERN CONTINUES]

Page No. _____
Date 31 08 23

Q1. Print Full Pyramid Pattern.

→ Represent Space



Rule:-

1. No. of Rows = n = 5
   ↳ Outer Loop = 5 Times

2. Write, what is printing in each row.

   r_0 → 4 spaces, 1★
   r_1 → 3 spaces, 2★
   r_2 → 2 spaces, 3★
   r_3 → 1 space, 4★
   r_4 → 0 space, 5★

→ formula for stars

row + 1

Dry Run:- n = 5,
r_0 → 0 + 1 = 1★
r_1 → 1 + 1 = 2★
r_2 → 2 + 1 = 3★
r_3 → 3 + 1 = 4★
r_4 → 4 + 1 = 5★

→ formula for spaces

n - row - 1

No. of Rows ↵ ↳ value of Current Row

Dry Run → n = 5,
r_0 → 5 - 0 - 1 = 4 spaces
r_1 → 5 - 1 - 1 = 3 spaces
r_2 → 5 - 2 - 1 = 2 spaces
r_3 → 5 - 3 - 1 = 1 space
r_4 → 5 - 4 - 1 = 0 space

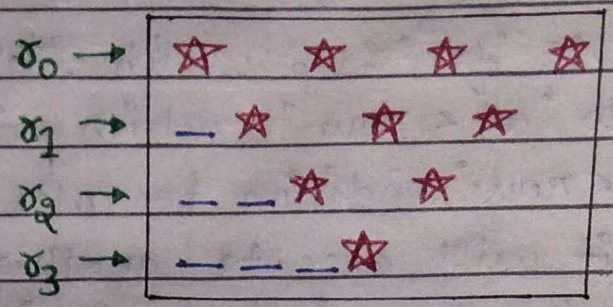
Code :-

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

Flow of execution :-

- for every value of row, from row='0' to 'row<n' where 'n' is total no. of rows, inner loops will executes.
  - ↳ for col='0' to 'col < n-row-1', it will print space ' '.
    - In each row, it will print 'n-row-1' times space and terminates the execution when condition becomes false.
  - ↳ After printing spaces, it will print stars '* _' from col='0' to 'col < row+1'.
    - It will print 'row+1' times stars in each row.

Q.2. Print Inverted Full Pyramid Pattern.



_ → Represent Space

Rule :-

1. No. of Rows = 4 = n
   ↳ Outer Loop = 4 Times

2. r_0 → 0 space, 4☆
   r_1 → 1 space, 3☆
   r_2 → 2 space, 2☆
   r_3 → 3 space, 1☆

→ Formula for space
r times space
↳ Current Row no.

Dry Run :- n = 4 (Total no. of Rows)
r_0 → 0 space
r_1 → 1 space
r_2 → 2 space
r_3 → 3 space

→ formula for star
n - row
Total no. of Rows ↵ ↳ Current Row

Dry Run :- n = 4
r_0 → 4 - 0 = 4☆
r_1 → 4 - 1 = 3☆
r_2 → 4 - 2 = 2☆
r_3 → 4 - 3 = 3☆

Code :-

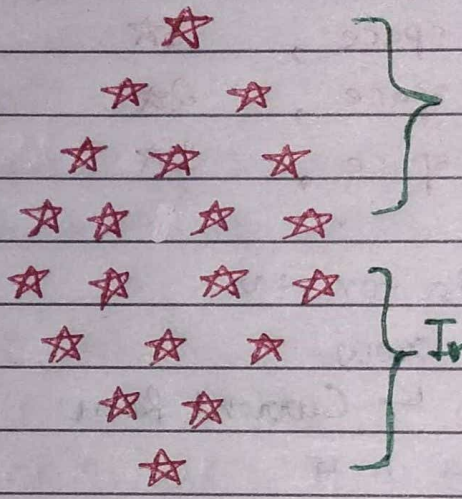
```
for( int row=0; row<n; row=row+1) {
    for ( int col=0; col<row; col=col+1) {
        cout << " ";
    }
    for( int col=0; col<n-row; col=col+1){
        cout << "☆ ";
    }
    cout << endl;
}
```

Page No.
Date

flow of execution:- Total no. of Rows

- for every iteration of 'row' from '0' to 'n', it will execute 'for' loop which iterates from 'col = 0' to 'col < row' which print a space until 'col < row' condition becomes false.
- After that it will execute another for loop, which executes from 'col = 0' to 'col < n - row' and print a star until 'col < n - row' condition becomes false.

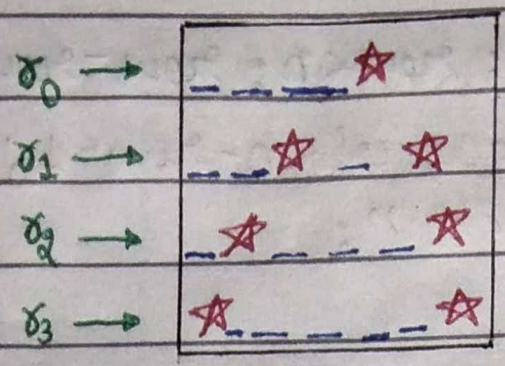
Q3. Print Diamond Pattern.



- for this pattern, we can use full pyramid pattern for creating upper half of the diamond pattern and use inverted full pyramid to create lower half of the diamond pattern.
- Just keep this in mind that, we have to divide total no. of rows by 2. To get overall diamond pattern within specified no. of rows.

Otherwise, it will create 'n' no. of rows for upper half and 'n' no. of rows for lower half which creates twice no. of total rows for diamond pattern.

Q4. Print Hollow Pyramid Pattern.



Rule:-

1. Total no. of Rows = 4 = n
   ↳ Outer Loop = 4 Times

2. r0 → 3 space , 1 ★
   r1 → 2 space , 1 ★ , 1 sp , 1 ★
   r2 → 1 space , 1 ★ , 3 space , 1 ★
   r3 → 0 space , 1 ★ , 5 space , 1 ★

→ Formula for printing space outside the pyramid.

n - row - 1
↵ ↳ value of current row
Total no. of Rows

Dry run :- n = 4

r0 → 4 - 0 - 1 = 3 space
r1 → 4 - 1 - 1 = 2 space
r2 → 4 - 2 - 1 = 1 space
r3 → 4 - 3 - 1 = 0 space

→ Formula for printing hollow pyramid.

for stars → row + 1 (Only if col = 0 or col = row)
Else print space other than col = 0 and col = row

code :-

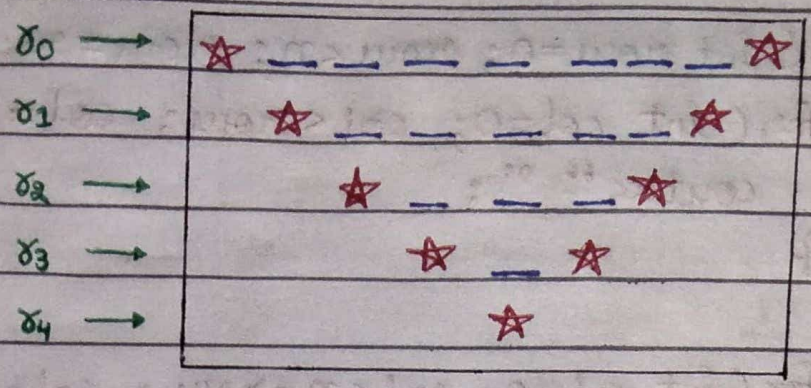
```
for ( int row=0; row<n; row=row+1) {
    for ( int col=0; col<n-row-1; col=col+1){
        cout << " ";
    }
    for ( int col=0; col<row+1; col=col+1){
        if ( col==0 || col== row){
            cout << "* ";
        }
        else {
            cout << "  ";
        }
    }
    cout << endl;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

Flow of execution :-

- Outer loop will iterate for each row from 'row=0' to 'row<n', and executes until 'row<n' becomes false.

- for every iteration of col from '0' to 'row+1' it will print * and space according to conditions-
  - if col=0 or col=row, then it will print '*'
  - else, it will print ' ' (space).

Q5. Print inverted hollow pyramid.



Rule :-

1. No. of Rows = 5
   ↳ Outer Loop = 5 Times

2. r0 →
   r1 →
   r2 →
   r3 →
   r4 →

   Each row has ☆ on its first and last column. And other columns between them have spaces.

→ formula for space outside the pattern :-

row (times space)

r0 → 0 space
r1 → 1 space
r2 → 2 space
r3 → 3 space
r4 → 4 space

→ formula for printing hollow pyramid pattern :-

n - row
Total Rows ↵ ↳ Current Row

↳ for printing stars :-
- if col = 0 or col = n - row - 1
- else, print space

Code:-

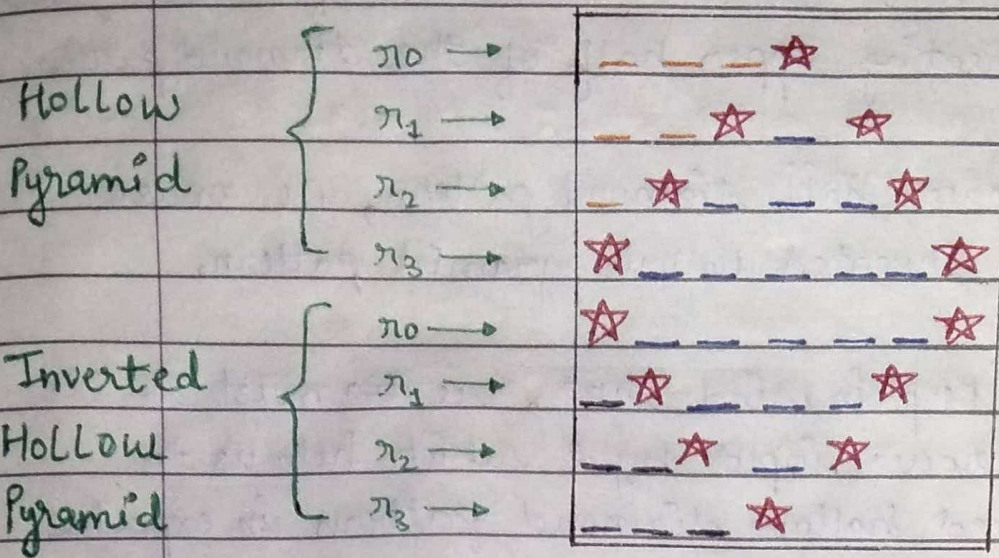
```
for ( int row=0; row<n; row=row+1){
   for( int col=0; col<row; col=col+1){
      cout<<" ";
   }

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

Flow of execution:-

- Outer loop will iterate from row='0' to 'row<n'.
- first inner loop will iterate from col='0' to 'col<row' and print outer spaces.
- Second inner loop will iterate from col='0' to 'col< n-row' and print '*' if 'col=0' or 'col = n-row-1'.
  else, it will print space.

Page No.
Date

Q6. Print Hollow Diamond Pattern.



↳ Represents space outside in Upper Half

↳ Represents space Outside in Lower Half

↳ Represents space inside the pattern

Rule:-

1. Total no. of Rows = n = 8.
   ↳ Outer Loop = 8 Times

2. r_0 → 3 space, 1★
   r_1 → 2 space, 1★, 1 sp, 1★
   r_2 → 1 space, 1★, 3 sp, 1★
   r_3 → 0 space, 1★, 5 sp, 1★
   r_4 → 0 space, 1★, 5 sp, 1★
   r_5 → 1 space, 1★, 3 sp, 1★
   r_6 → 2 space, 1★, 1 sp, 1★
   r_7 → 3 space, 1★

→ formula for spaces outside in Upper Half:-

n = n/2 :- , n - row - 1 , n = 8/2 = 4

r_0 → 4 - 0 - 1 = 3 spaces
r_1 → 4 - 1 - 1 = 2 spaces
r_2 → 4 - 2 - 1 = 1 space
r_3 → 4 - 3 - 1 = 0 space

→ formula for ★ in Upper Half

row + 1 , n = 4

r_0 = 0 + 1 = 1★
r_1 = 1 + 1 = 2★
r_2 = 2 + 1 = 3★
r_3 = 3 + 1 = 4★

Only (c = 0 or c = row)
↓ col ↓ col

→ formula for ★ in Lower Half

n = 4, n - row

r_0 → 4 - 0 = 4★
r_1 → 4 - 1 = 3★
r_2 → 4 - 2 = 2★
r_3 → 4 - 3 = 1★

Only (col = 0 or col = n - row - 1)

→ Formula for spaces Outside in Lower Half:-

n = n/2, n = 8/2 = 4

row (Times Space)

r_0 → 0 space
r_1 → 1 space
r_2 → 2 space
r_3 → 3 space

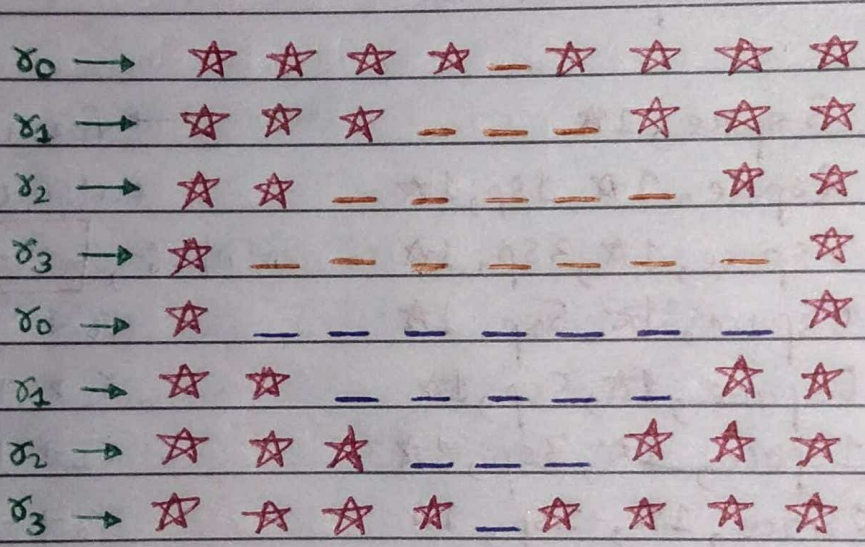
Code :-

- for this pattern we can use exact code of Hollow Pyramid pattern for creating upper half of the diamond pattern.

- for creating lower half diamond pattern, we can use exact code of Inverted Hollow pyramid pattern.

- We have to keep in mind that, we must divide total no. of rows input by 2 which help us to get the desired hollow diamond pattern in expected no. of rows.

Q7. Print Flipped Solid Diamond Pattern.



Rule :-

1. Total no. of Rows :- = 8 ; n = n/2 = 4
   ↳ Outer loop = 4 Times [for both Upper & Lower Halves]

2. Upper Half :-
   - r0 → 4★, 1 space, 4★
   - r1 → 3★, 3 space, 3★
   - r2 → 2★, 5 space, 2★
   - r3 → 1★, 7 Space, 1★

Page No.
Date

Dry Run
$r_0 \rightarrow 4-0=4$ ★
$r_1 \rightarrow 4-1=3$ ★
$r_2 \rightarrow 4-2=2$ ★
$r_3 \rightarrow 4-3=1$ ★
n=4

→ formula for printing ★ in Upper Half
n-row

→ formula for printing Space in Upper Half
2*row+1

Dry Run (n=4)
$r_0 \rightarrow 2\times0+1=1$ sp
$r_1 \rightarrow 2\times1+1=3$ sp
$r_2 \rightarrow 2\times2+1=5$ space
$r_3 \rightarrow 2\times3+1=7$ space

Lower Half:-
$r_0 \rightarrow$ 1★, 7space, 1★
$r_1 \rightarrow$ 2★, 5space, 2★
$r_2 \rightarrow$ 3★, 3space, 3★
$r_3 \rightarrow$ 4★, 1space, 4★

→ formula for printing ★ in Lower Half
row+1
n=4

Dry Run
$r_0 \rightarrow 0+1=1$★
$r_1 \rightarrow 1+1=2$★
$r_2 \rightarrow 2+1=3$★
$r_3 \rightarrow 3+1=4$★

→ Formula for printing Space in Lower Half
2*n-2*row-1
n=4

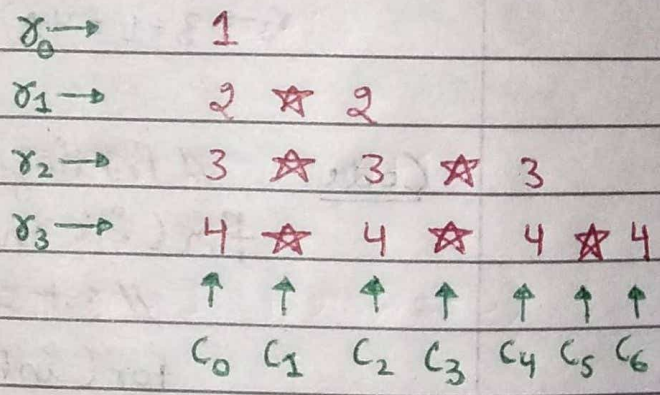
Dry Run
$r_0 \rightarrow 2\times4-2\times0-1=7$
$r_1 \rightarrow 2\times4-2\times1-1=5$
$r_2 \rightarrow 2\times4-2\times2-1=3$
$r_3 \rightarrow 2\times4-2\times3-1=1$

Code:-

```
// Printing Upper Half
for(int row=0; row<n; row=row+1){
    // 1st Inverted★ pyramid
    for(int col=0; col<n-row; col=col+1){
        cout<<"★";
    }
    // Full Pyramid 1
    for(int col=0; col<2*row+1; col=col+1){
        cout<<" ";
    }
    // 2nd Inverted★ Pyramid
    for(int col=0; col<n-row; col=col+1){
        cout<<"★";
    }
    cout<<endl;}
```

```
// Printing Lower Half
for (int row=0; row<n; row=row+1) {
// Full Pyramid 1
for (int col=0; col<row+1; col=col+1) {
        cout << "*";
    }
// Inverted Full pyramid
for (int col=0; col<2*n-2*row-1; col=col+1){
        cout << " ";
    }
// Full pyramid 2
for (int col=0; col<row+1; col=col+1){
        cout << "*";
    }
  cout << endl;
}
```

Q8.



Rule

1. Total no. of Rows = n = 4
   ↳ Outer loop = 4 Times

2. r_0 → 1 character
   r_1 → 3 character
   r_2 → 5 character
   r_3 → 7 character
   } 2*row+1 = Inner loop
   ↳ if col%2=0 → row+1
   ↳ else → *

Code:-

```
for( int row=0; row<n; row=row+1 ){
    for(int col=0; col<2*row+1; col=col+1){
        if (col%2==0){
            cout << row+1;
        }
        else {
            cout << "*";
        }
    }
    cout << endl;
}
```

Same as above

```
1
2 * 2
3 * 3 * 3
4 * 4 * 4 * 4
```

HW

Rules:-

1. Here, n=4 but only 3 rows are printing.
   ↳ Outerloop = row<n Times

```
r0 → 3 * 3 * 3
r1 → 2 * 2
r2 → 1
```

} HW

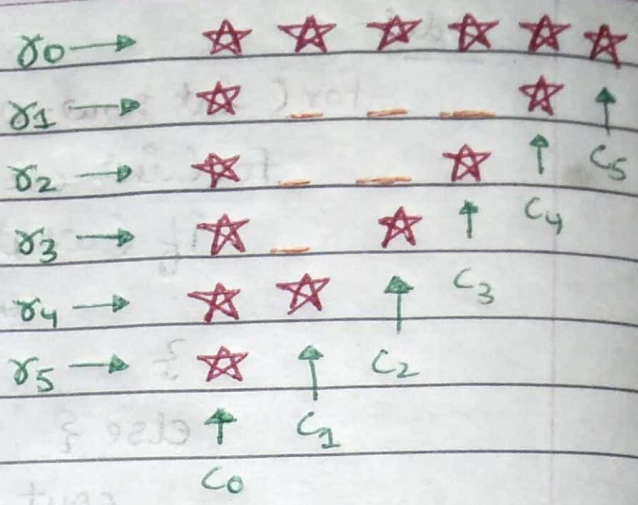
2. r0 → 5 character
   r1 → 3 character
   r2 → 1 character

   } 2*n-2*row-3 = Innerloop
   ↳ if (col%2=0)
   ↳ Print → n-row-1
   ↳ else
   ↳ Print → "*"

Code:-

```
for (int row=0; row<n; row=row+1)
 {
  for ( int col=0; col<2*n-2*row-3; col++)
  {
    if ( col%2==0){
        cout << n-row-1;
    }
    else{
        cout<<"*";
    }
}
cout<<endl; }
```

Q 9.



Rule :-

1. No. of rows = n = 6.
   ↳ Outer Loop = 6 Times

2. r_0 → 6 character
   r_1 → 5 character
   r_2 → 4 character
   r_3 → 3 character
   r_4 → 2 character
   r_5 → 1 character

   Inner Loop = n - row
   ↳ if((col=0 || col=n-row-1) || row=0)
   ↳ print ☆
   ↳ else
   ↳ print space

→ We only want ☆ in first row and First and last column of each row.

Code :-

```
for (int row=0; row<n; row=row+1)
{
    for( int col=0; col < n-row; col = col+1)
    {
      if(( col==0 || col==n-row-1) || row==0)
        cout << "☆";
      else
        cout << "_";
    }
    cout << endl;
}
```

Date

Q10.

$r_0 \rightarrow$ A
$r_1 \rightarrow$ A B A
$r_2 \rightarrow$ A B C B A
$r_3 \rightarrow$ A B C D C B A
$r_4 \rightarrow$ A B C D E D C B A

Rule:-

1. No. of Rows = n = 5
   ↳ Outer Loop = 5 Times

2. $r_0 \rightarrow$ A
   $r_1 \rightarrow$ A B
   $r_2 \rightarrow$ A B C
   $r_3 \rightarrow$ A B C D
   $r_4 \rightarrow$ A B C D E
   } Pattern Excluding Repetition of characters
   ↳ Inner Loop = row+1 (Times)

How to print characters?

Ex:- char ch = 'A' + 1 - 1 = A

∴ If we add same value in any variable and then subtract the same value then it won't affect the original value of that variable.

Similarly,

n + 'A' - 1

- n: Any given number
- 'A': character to be mapped
- 1: Subtract the no. from which we want to start mapping

Code:-

```
for(int row=0; row<n; row=row+1){
  char ch;
  for(int col=0; col<row+1; col=col+1){
    int number = col+1;
    ch = number + 'A' - 1;
    cout << ch;
  }
  // printing characters in backward direction until we reach 'A'
```

```
        for (char alphabet = ch; alphabet > 'A'; )
        {
            alphabet = alphabet - 1;
            cout << alphabet;
        }
        cout << endl;
    }
```

Flow of execution :-

- Outer loop will iterate until 'row<n' becomes false.
  - ↳ first for loop will iterate from 'col=0' to 'col<row+1' and print character values starting from 'A'.
  - ↳ Second for loop will iterate from 'alphabet = ch' to alphabet > 'A' and first decrease the value of alphabet by 1 and then print the alphabet value to get the previous value of 'ch' every time until it get 'A'.

Homework Questions

Q1. Numeric hollow inverted half pyramid.

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

n=5

```
1 2 3 4 5
2 _ _ 5
3 _ 5
4 5
5
```

Q2. Numeric palindrome equilateral pyramid.

```
for (int row=0 ; row<n; row =row+1)
{
  // first print n-row-1 spaces
  for (int col=0; col<n-row-1; col++){
     cout<< " ";
  }
  // Print half pyramid patterns of numbers
   int num;
   for (int col=0; col< row+1; col=col +1) {
       int number= col+1;
       num = number+1-1 ;
       cout<<num;    // simply print numeric pyramid
   }
   // Now print repeating numbers in reverse order until 1
    for (int col = num; col >1 ;){
        col = col-1;
        cout<<col;
    }
  cout <<endl;
}
```

```
        1
      1 2 1
    1 2 3 2 1
  1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```