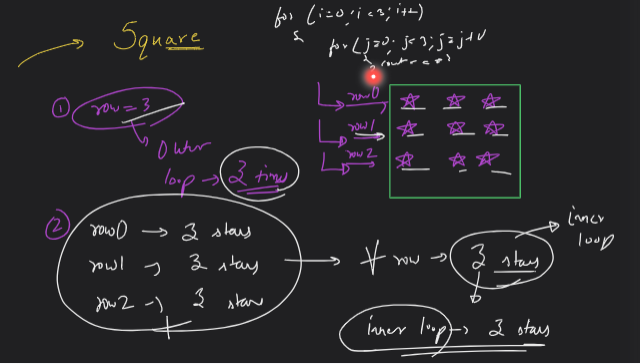
**PATTERNS**

Pattern printing rules

1. Find the number of rows
2. Write down what is happening in each row eg:-



void squarePattern()

{

    for (int i = 0; i < 4; i++)

    {

        for (int j = 0; j < 4; j++)

        {

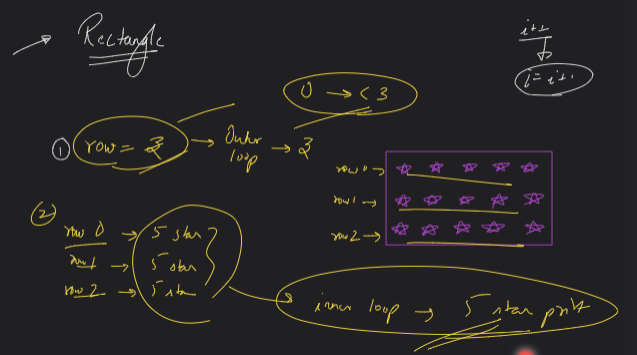
            cout << "\* ";

        }

        cout << endl;

    }

}



void rectangle()

{

    for (int j = 0; j < 3; j++)

    {

        for (int i = 0; i < 5; i++)

        {

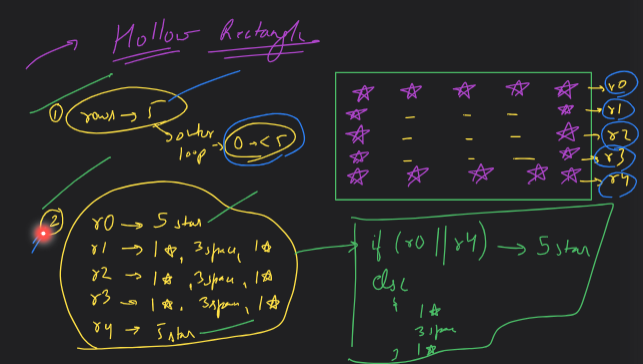
            cout << "\* ";

        }

        cout << endl;

    }

}



Outer loop -> no. of rows

Inner loop -> no. of columns

void hollow\_rectangle()

{

    for (int row = 0; row < 5; row++)

    {

        for (int col = 0; col < 5; col++)

        {

            if (row == 0 || row == 4)

            {

                cout << "\* ";

            }

            else

            {

                if (col == 0 || col == 4)

                {

                    cout << "\* ";

                }

                else

                {

                    cout << "  ";

                }

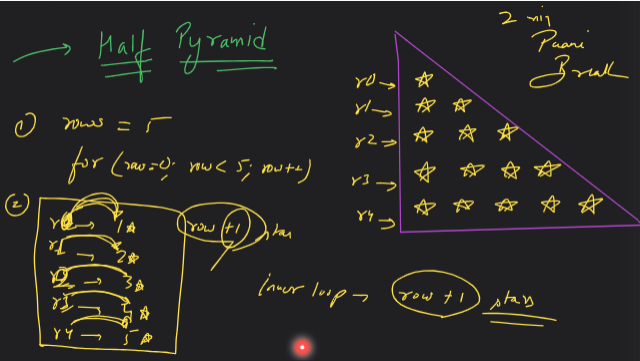
            }

        }

        cout << endl;

    }

}



void halfPyramid()

{

    for (int row = 0; row < 5; row++)

    {

        for (int col = 0; col < row + 1; col++)

        {

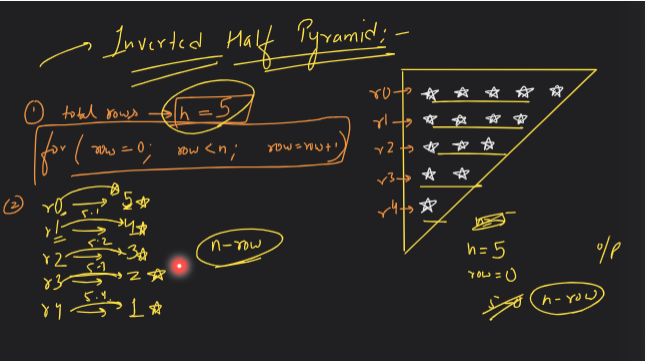
            cout << "\* ";

        }

        cout << endl;

    }

}



void inverted\_halfPyramnid()

{

    for (int row = 0; row < 5; row++)

    {

        for (int col = 0; col < 5 - row; col++)

        {

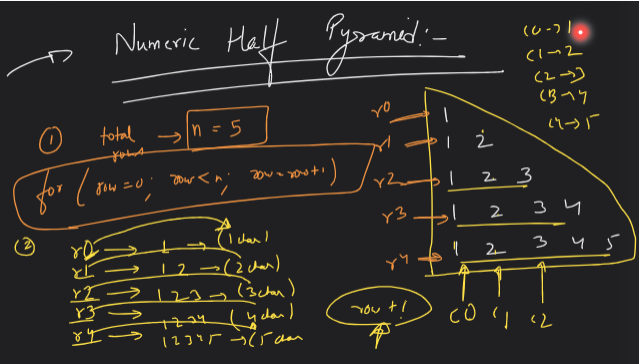
            cout << "\* ";

        }

        cout << endl;

    }

}



void numric\_halfPyramid()

{

    for (int row = 0; row < 5; row++)

    {

        for (int col = 0; col < row + 1; col++)

        {

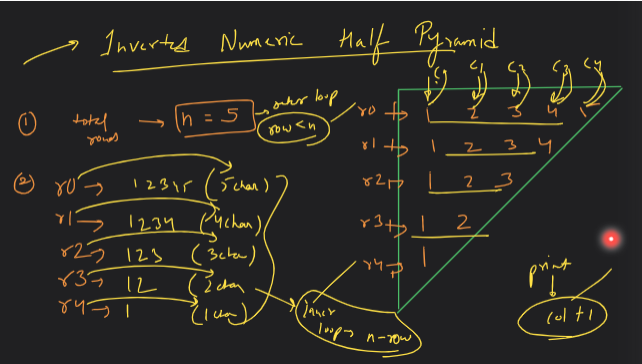
            cout << col + 1 << " ";

        }

        cout << endl;

    }

}



void InvertedNumeric\_halfPyramid()

{

    for (int row = 0; row < 5; row++)

    {

        for (int col = 0; col < 5 - row; col++)

        {

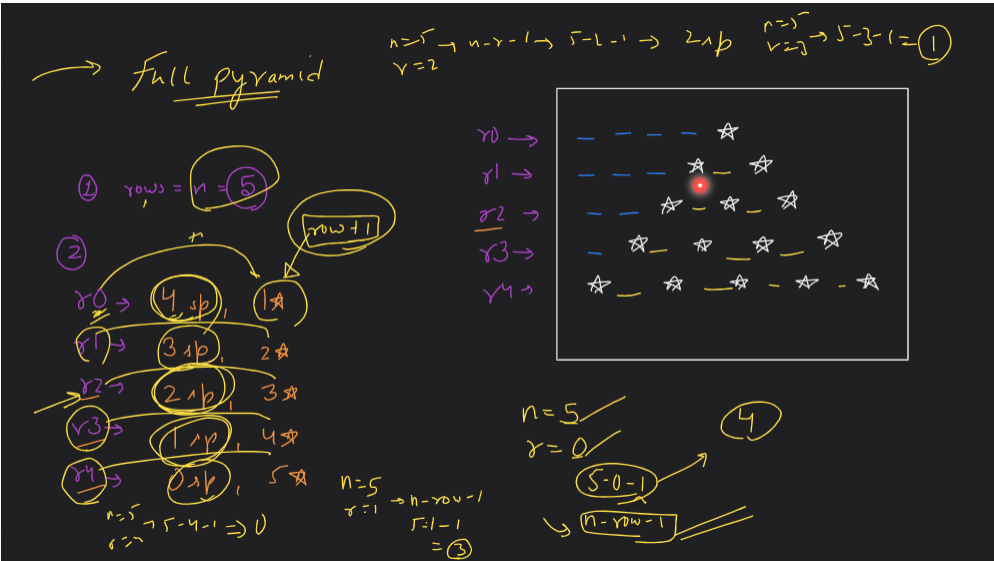
            cout << col + 1 << " ";

        }

        cout << endl;

    }

}



void fullPyramid()

{

    for (int row = 0; row < 5; row++)

    {

        for (int col = 0; col < 5 - row - 1; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < row + 1; col++)

        {

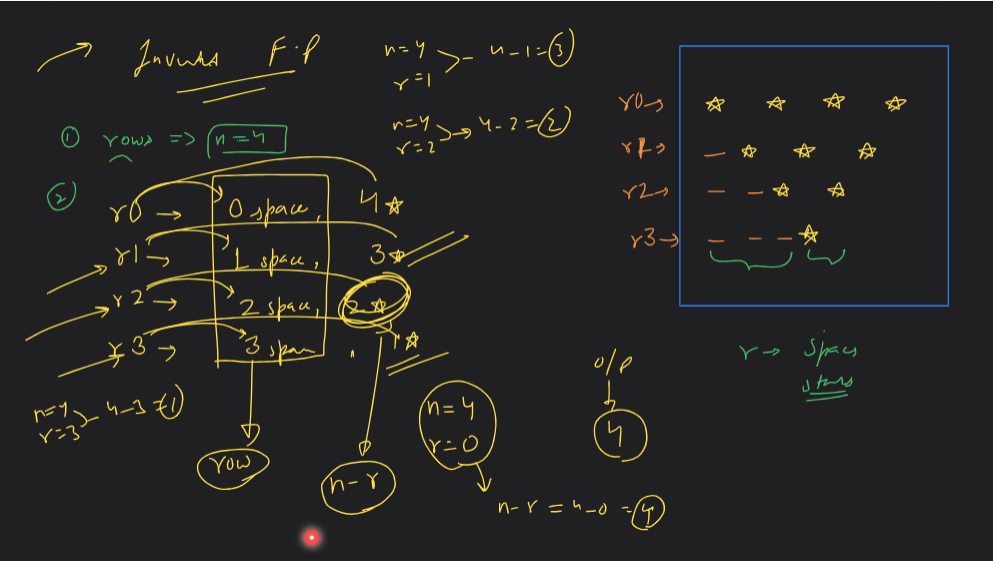
            cout << "\* ";

        }

        cout << endl;

    }

}



void invertedFullPyramid()

{

    for (int row = 0; row < 4; row++)

    {

        for (int col = 0; col < row; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < 4 - row; col++)

        {

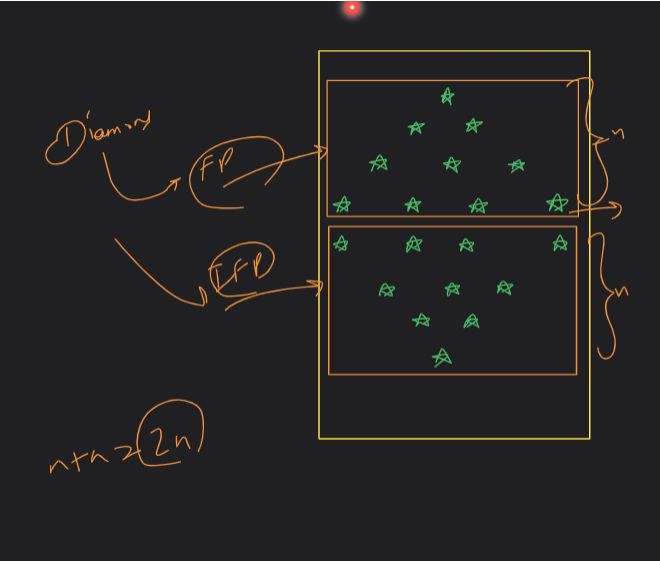
            cout << "\* ";

        }

        cout << endl;

    }

}



void diamond(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < n - row - 1; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < row + 1; col++)

        {

            cout << "\* ";

        }

        cout << endl;

    }

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < row; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < n - row; col++)

        {

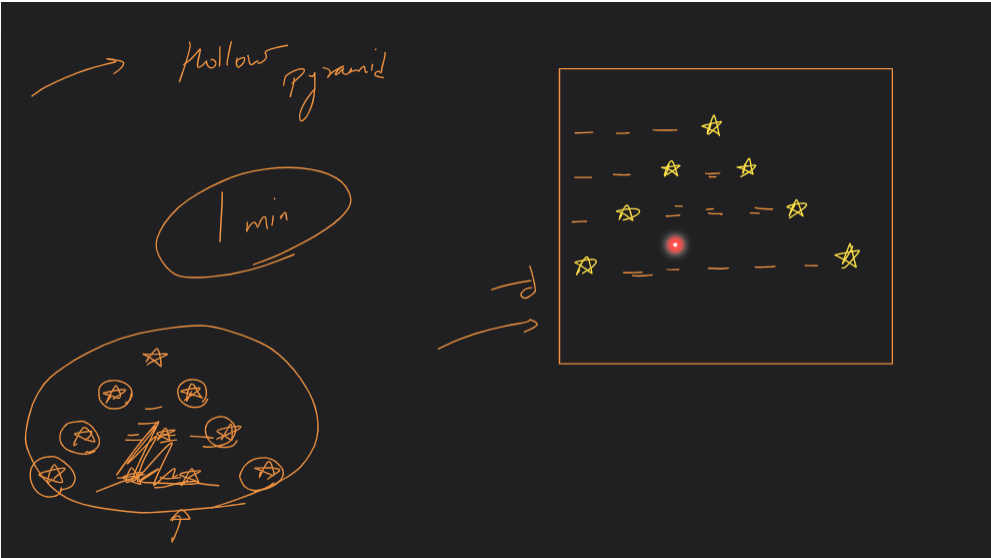
            cout << "\* ";

        }

        cout << endl;

    }

}



void hollow\_pyramid()

{

    for (int row = 0; row < 20; row++)

    {

        for (int col = 0; col < 20 - row - 1; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < row + 1; col++)

        {

            if (col == 0 || col == row)

            {

                cout << "\* ";

            }

            else

            {

                cout << "  ";

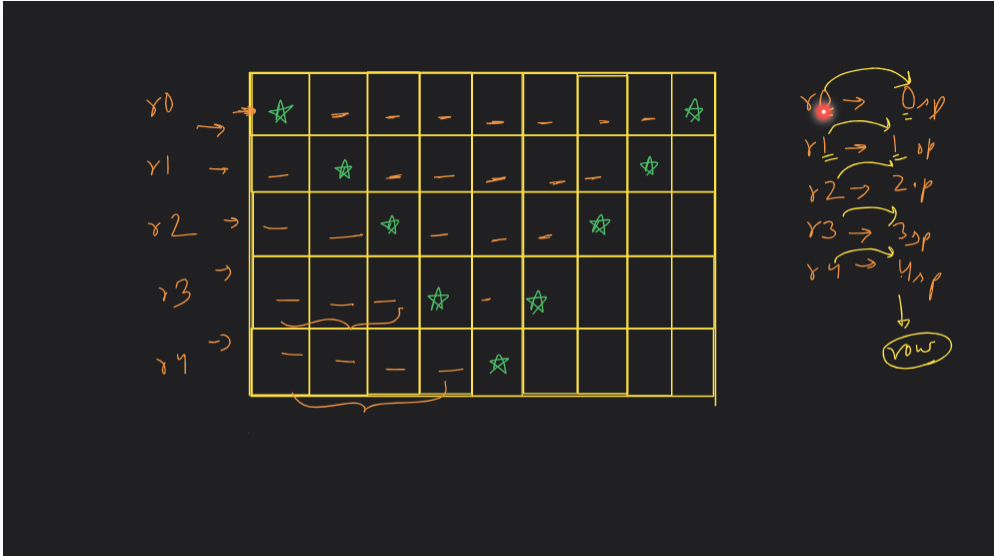
            }

        }

        cout << endl;

    }

}



void inverted\_hollow\_pyramid(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < row; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < n - row; col++)

        {

            if (col == 0 || col == n - row - 1)

            {

                cout << "\* ";

            }

            else

            {

                cout << "  ";

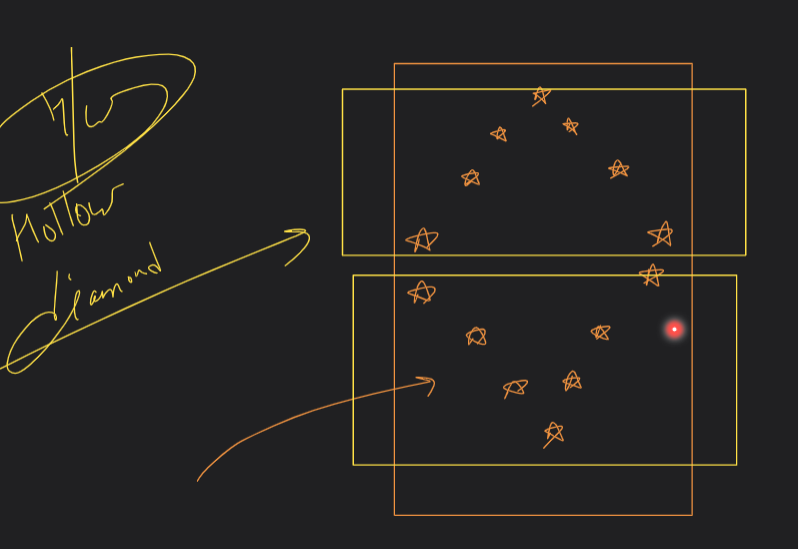
            }

        }

        cout << endl;

    }

}



void hollow\_diamond(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < n - row - 1; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < row + 1; col++)

        {

            if (col == 0 || col == row)

            {

                cout << "\* ";

            }

            else

            {

                cout << "  ";

            }

        }

        cout << endl;

    }

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < row ; col++)

        {

            cout << " ";

        }

        for (int col = 0; col < n - row; col++)

        {

            if (col == 0 || col == n - row - 1)

            {

                cout << "\* ";

            }

            else

            {

                cout << "  ";

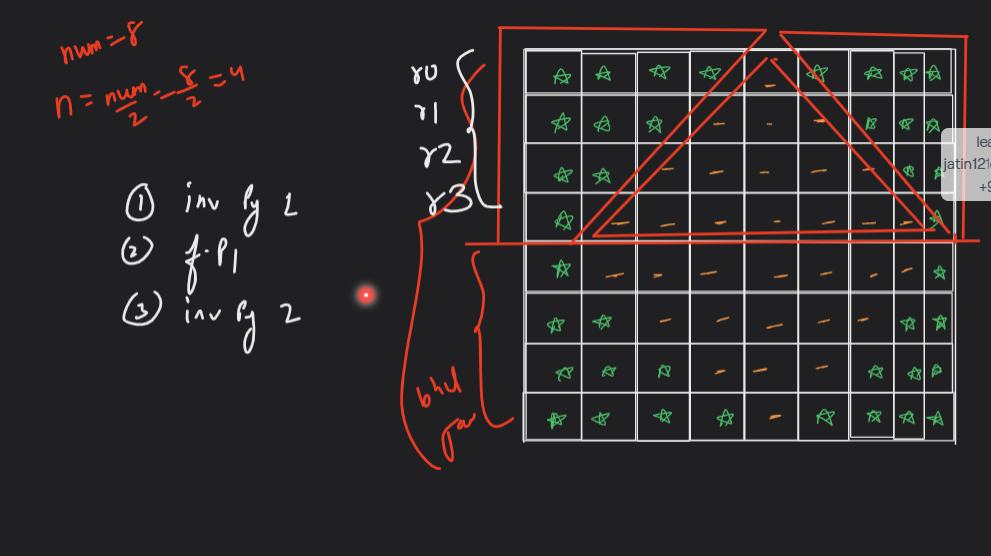
            }

        }

        cout << endl;

    }

}



void complex(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < n - row ; col++)

        {

            cout << "\* ";

        }

        for (int col = 0; col < ((2 \* row) + 1); col++)

        {

            cout << "  ";

        }

        for (int col = 0; col < n - row; col++)

        {

            cout << "\* ";

        }

        cout << endl;

    }

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < row + 1; col++)

        {

            cout << "\* ";

        }

        for (int col = 0; col < ((2 \* n) - 1 - (2 \* row)); col++)

        {

            cout << "  ";

        }

        for (int col = 0; col < row + 1; col++)

        {

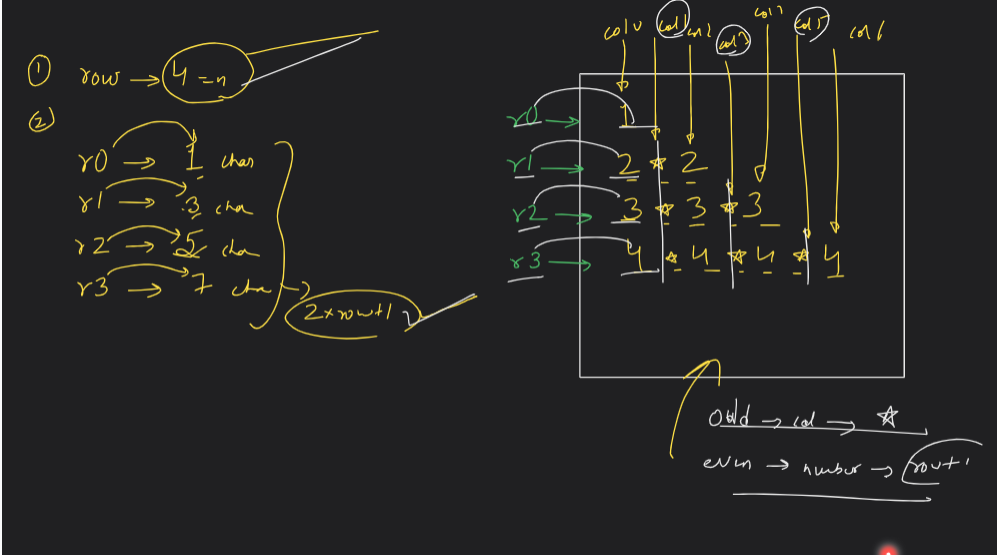
            cout << "\* ";

        }

        cout << endl;

    }

}



void num\_star(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < (2 \* row) + 1; col++)

        {

            if ((col % 2) == 0)

            {

                cout << row + 1 << " ";

            }

            else

            {

                cout << "\* ";

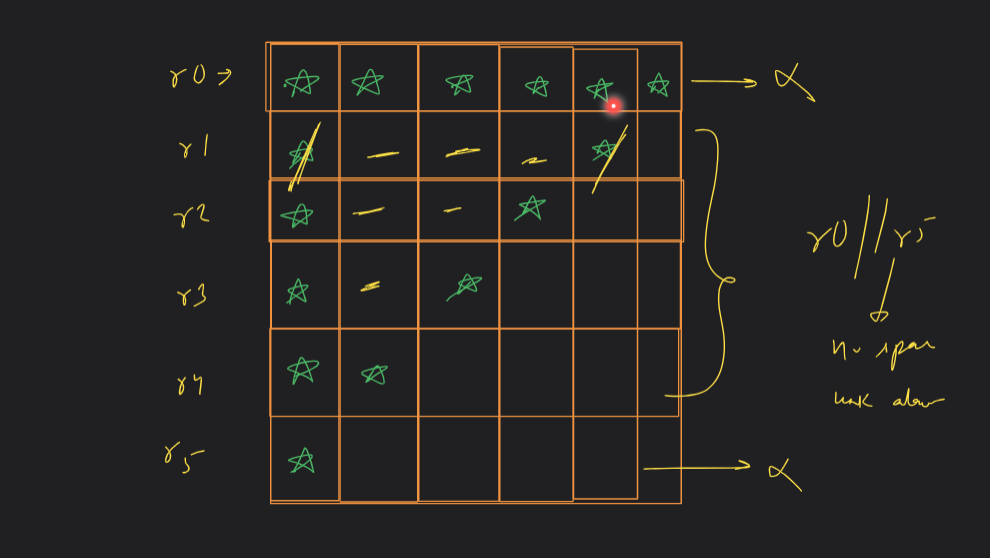
            }

        }

        cout << endl;

    }

}



void inverted\_hollow\_halfPyramid(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < n - row; col++)

        {

            if (row == 0 || row == n - 1)

            {

                cout << "\* ";

            }

            else

            {

                if (col == 0 || col == n - row - 1)

                {

                    cout << "\* ";

                }

                else

                {

                    cout << "  ";

                }

            }

        }

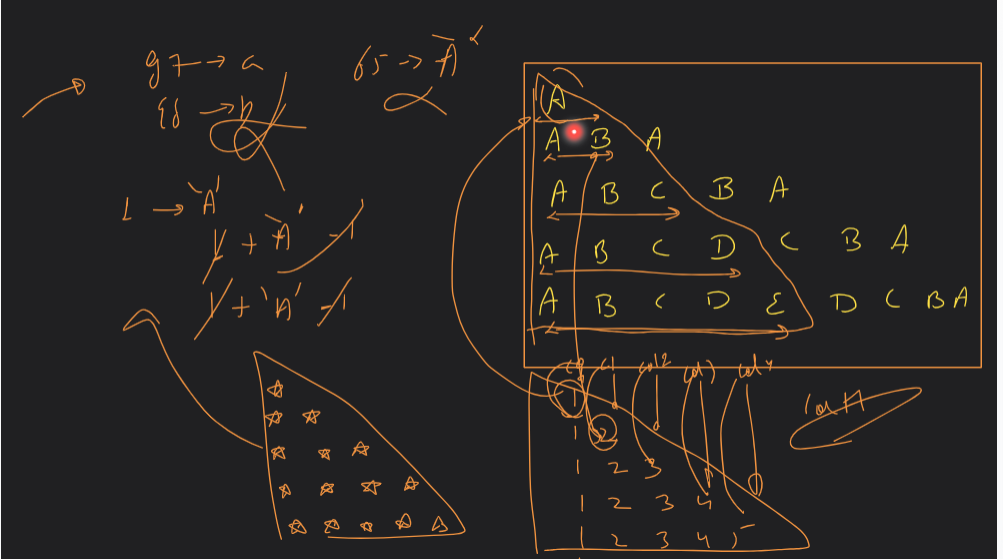
        cout << endl;

    }

}

To convert 1 into A

1 -> 1 + ‘A’ – 1



void charPattern(int n)

{

    char ch;

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < row + 1; col++)

        {

            ch = col + 1 + 'A' - 1;

            cout << ch<<" ";

        }

        for (char i = ch; i > 'A';)

        {

            i = i - 1;

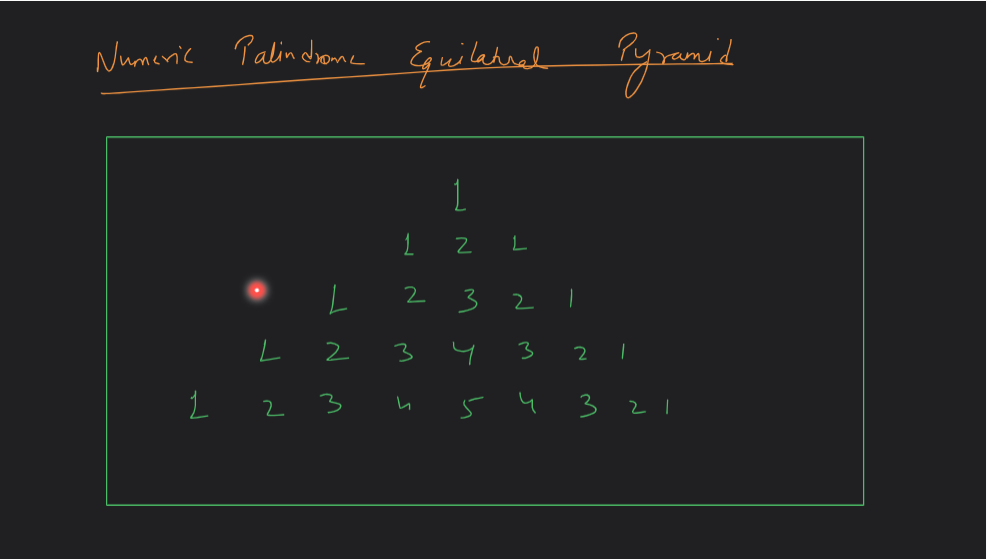
            cout << i << " ";

        }

        cout << endl;

    }

}



void Numeric\_Palindrome\_equilateral\_Pyramid(int n)

{

    for (int row = 0; row < n; row++)

    {

        int k;

        for (int col = 0; col < row + 1; col++)

        {

            k = col + 1;

            cout << k << " ";

        }

        for (int i = k - 1; i > 0; i--)

        {

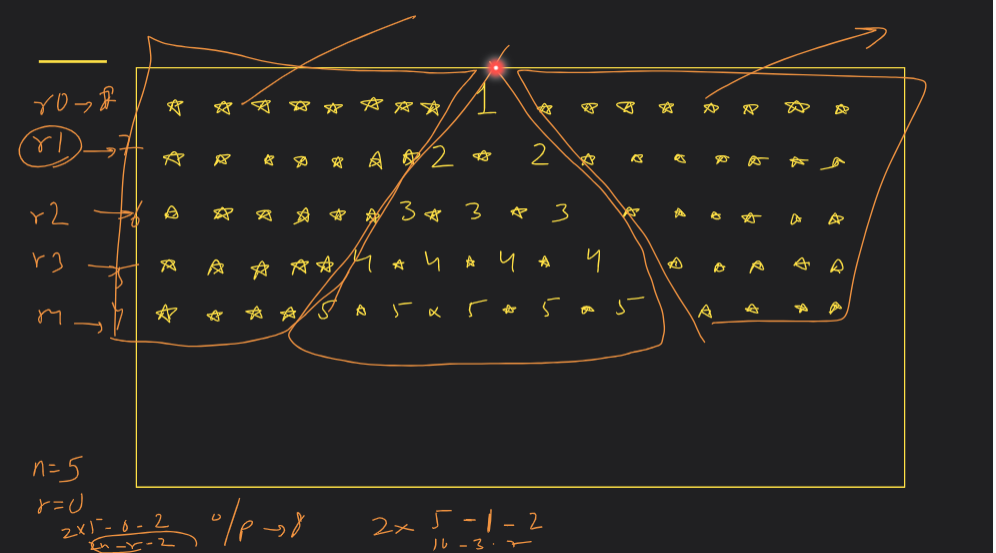
            cout << i << " ";

        }

        cout << endl;

    }

}



void complex(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < (2 \* n - row - 2); col++)

        {

            cout << "\* ";

        }

        for (int col = 0; col < (2 \* row) + 1; col++)

        {

            if ((col % 2) == 0)

            {

                cout << row + 1 << " ";

            }

            else

            {

                cout << "\* ";

            }

        }

        for (int col = 0; col < (2 \* n - row - 2); col++)

        {

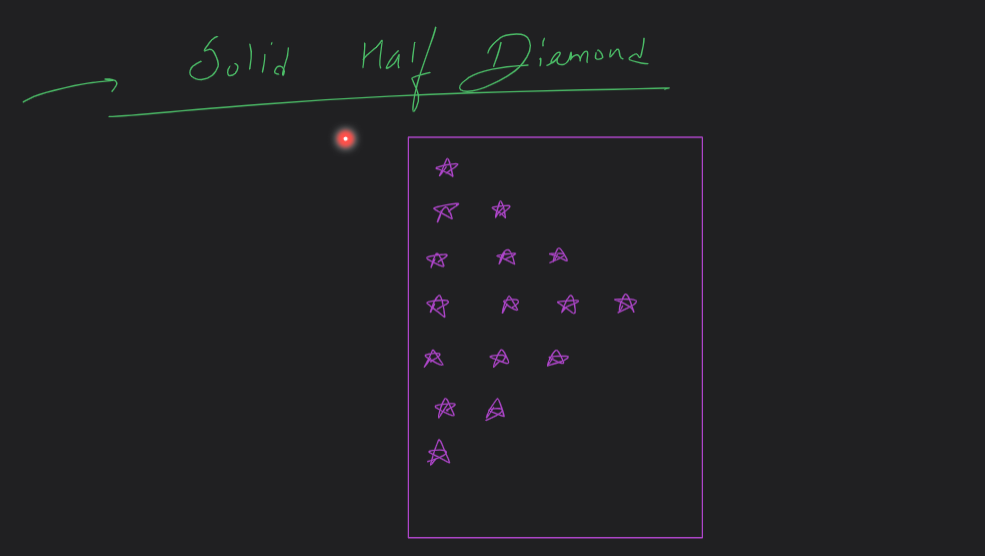
            cout << "\* ";

        }

        cout << endl;

    }

}



void Solid\_Half\_diamond(int n)

{

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < row + 1; col++)

        {

            cout << "\* ";

        }

        cout << endl;

    }

    for (int row = 1; row < n; row++)

    {

        for (int col = 0; col < n - row; col++)

        {

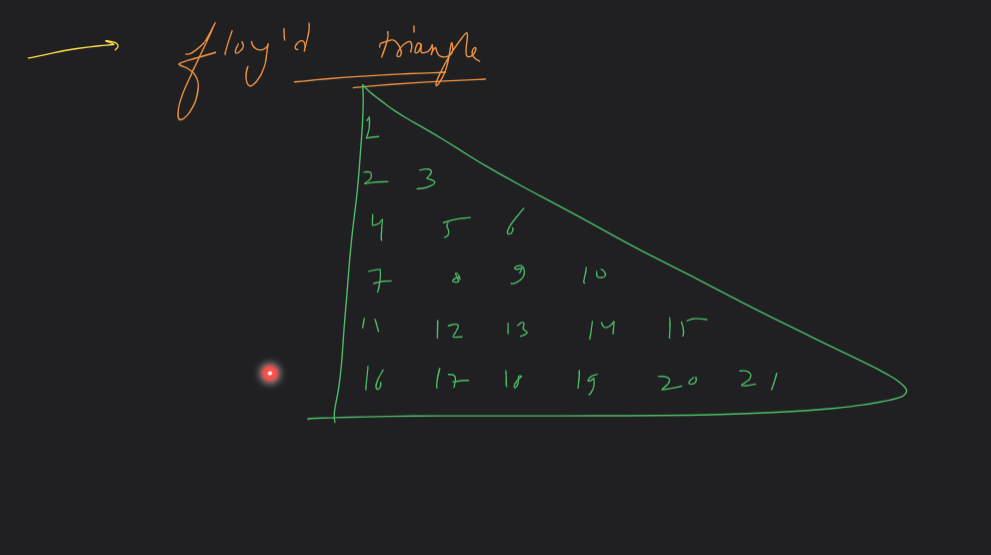
            cout << "\* ";

        }

        cout << endl;

    }

}



void floyyd\_traingle(int n){

    int k = 1;

    for(int row = 0; row< n;row++){

        for(int col = 0 ; col< row + 1;col++){

            cout<< k<<" ";

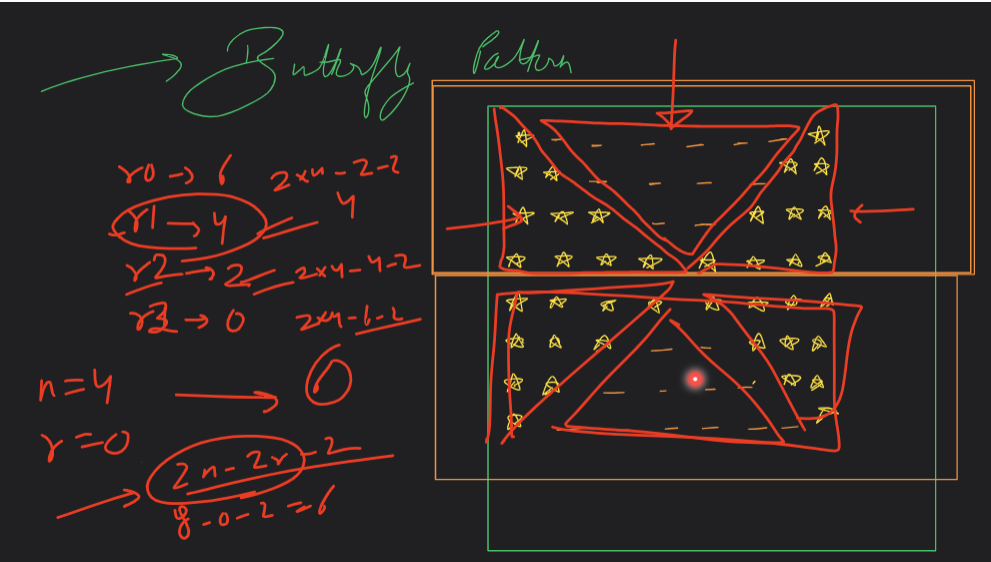
            k++;

        }

        cout<<endl;

    }

}



void butterfly\_pattern(int n)

{

    int n\_ = n / 2;

    for (int row = 0; row < n\_; row++)

    {

        for (int col = 0; col < row + 1; col++)

        {

            cout << "\* ";

        }

        for (int col = 0; col < 2 \* n\_ - 2 \* row - 2; col++)

        {

            cout << "  ";

        }

        for (int col = 0; col < row + 1; col++)

        {

            cout << "\* ";

        }

        cout << endl;

    }

    for (int row = 0; row < n\_; row++)

    {

        for (int col = 0; col < n\_ - row; col++)

        {

            cout << "\* ";

        }

        for (int col = 0; col < 2 \* row; col++)

        {

            cout << "  ";

        }

        for (int col = 0; col < n\_ - row; col++)

        {

            cout << "\* ";

        }

        cout << endl;

    }

}