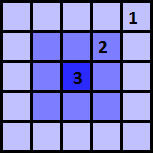
Let's assume that each square matrix consists of several *rings*.  
A matrix is *pyramid-like* if its outermost *ring* is filled with1s, the next *ring* is filled with 2s, the next one with3s, and so on.



Given a number N, return the *pyramid-like* matrix of size NxN.

**Example:**

For N = 1 the output should be

[[1]]

For N = 2 the output should be

[[1,1],

[1,1]]

For N = 3 the output should be

[[1,1,1],

[1,2,1],

[1,1,1]]

For N = 4 the output should be

[[1,1,1,1],

[1,2,2,1],

[1,2,2,1],

[1,1,1,1]]

* **[input] integer N**
  + Matrix size, 1 ≤ N ≤ 100.
* **[output] array.array.integer**
  + A *pyramid matrix*.

<https://codefights.com/challenge/CLtm7mPNrEJMYcfJN>

--ACEPTADO--

#include <iostream>

#include <conio.h>

#include <stdio.h>

#include <vector>

std::vector<std::vector<int> > pyramidMatrix(int N)

{

    int ret[N][ N];

    int fa=0,  cd=N-1,  fb = N-1, ci=0;

    int num = 1;

    while (ci <= cd)

    {

        for (int j = ci; j <= cd; j++)

        {

            ret[fa][ j] = num;

        }

        for (int i = fa; i <= fb; i++)

        {

            ret[i][cd] = num;

        }

        for (int j = ci; j <= cd; j++)

        {

            ret[fb][ j] = num;

        }

        for (int i = fa; i <= fb; i++)

        {

            ret[i][ci] = num;

        }

        num++;

        fa++;

        cd--;

        fb--;

        ci++;

    }

    std::vector<std::vector<int> > matriz; // = new List<List<int>>();

    for (int i = 0; i < N; i++)

    {

        std::vector<int> fila; // = new List<int>();

        for (int j = 0; j < N; j++)

        {

            fila.push\_back(ret[i][j]);

        }

        matriz.push\_back(fila);

    }

    return matriz;

}

int main(){

    std::vector<std::vector<int> > res = pyramidMatrix(6);

    for(int i =0; i < res.size(); i++)

    {

        for(int j = 0; j < res [i].size(); j++)

        {

           printf("%d ", res[i][j]);

        }

        printf("**\n**");

    }

    getch();

    return 0;

}