DSA With C++ and Java

L-1 Patterns

Pattern-1: Rectangular Star Pattern

Problem Statement: Given an integer **N**, print the following pattern.

Examples:

Example 1:

Input: N = 3

Output:

*** * ***

*** * ***

*** * ***

Example 2:

Input: N = 6

Output:

*** * * * * ***

*** * * * * ***

*** * * * * ***

*** * * * * ***

*** * * * * ***

*** * * * * ***

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question:

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.

- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns.

In this particular problem, we run the outer loop for N times since we have N rows to be printed, the inner loop also runs for N times as we have to print N stars in each row. This way we get a rectangular star pattern (square) with an equal number of rows and columns.

Code:

```
#include <iostream>

using namespace std;


void printSquarePattern(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cout << "* ";
        }
        cout << endl;
    }
}
int main() {
    int n;
    cout << "Enter the size of the square pattern: ";
    cin >> n;
    printSquarePattern(n);
    return 0;
}
```

Pattern-2: Right-Angled Triangle Pattern

Problem Statement: Given an integer **N**, print the following pattern :

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this problem, we run the outer loop for N times as we have to print N rows, and since we have to print a right-angled triangle/pyramid which must be upright, the inner loop will run for the row number in each iteration. For eg: 1 star for row 1, 5 stars for row 5, and so on.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern2(int N)
```

```
{
    // This is the outer loop which will loop for the rows.
    for (int i = 0; i < N; i++)
    {
        // This is the inner loop which loops for the columns
       // no. of columns = row number for each line here.
        for (int j = 0; j <=i; j++)
        {
            cout << "* ";
        }

        // As soon as stars for each iteration are printed, we move to the
        // next row and give a line break otherwise all stars
        // would get printed in 1 line.
        cout << endl;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern2(N);

    return 0;
}
```

Output

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

Pattern - 3: Right-Angled Number Pyramid

Problem Statement: Given an integer **N**, print the following pattern :

```
1
12
123
1234
12345
```

Here, N = 5.

Examples:

Input Format: N = 3

Result:

1

1 2

1 2 3

Input Format: N = 6

Result:

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

1 2 3 4 5 6

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this pattern, we run the outer loop for N times as we have to print N rows, and since we have to print a right-angled triangle/pyramid which must be upright, so the inner loop will run for the row number in each iteration. For eg: 1 number for row 1, 5 numbers for row 5, and so on. The only difference between this pattern and pattern 2 is that here we print **numbers** looping from 1 to the row number for each row instead of printing stars.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern3(int N)

{

    // This is the outer loop which will loop for the rows.

    for (int i = 1; i <= N; i++)

    {

        // This is the inner loop which loops for the columns

       // no. of columns = row number for each line here.

       // Here, we print numbers from 1 to the row number

       // instead of stars in each row.

        for (int j = 1; j <=i; j++)

        {

            cout <<j<<" ";

        }


        // As soon as numbers for each iteration are printed, we move to the

        // next row and give a line break otherwise all numbers

        // would get printed in 1 line.

        cout << endl;

    }

}


int main()

{

    // Here, we have taken the value of N as 5.

    // We can also take input from the user.

    int N = 5;


    pattern3(N);
```

```
    return 0;
}
```

Output

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

Pattern - 4: Right-Angled Number Pyramid - II

Problem Statement: Given an integer **N**, print the following pattern :

```
1
22
333
4444
55555
```

Here, N = 5.

Examples:

Input Format: N = 3

Result:

```
1
2 2
3 3 3
```

Input Format: N = 6

Result:

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
```

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this pattern, we run the outer loop for N times as we have to print N rows, and since we have to print a right-angled triangle/pyramid which must be upright, so the inner loop will run for the row number in each iteration. For eg: 1's for row 1, 5's 5 times for row 5, and so on. The only difference between this pattern and pattern 2 is that here we print numbers in each row instead of printing stars.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern4(int N)

{

    // This is the outer loop which will loop for the rows.

    for (int i = 1; i <= N; i++)

    {

        // This is the inner loop which loops for the columns

       // no. of columns = row number for each line here.

       // Here, we print numbers equal to the row number

       // instead of stars in each row.

        for (int j = 1; j <=i; j++)

        {

            cout <<i<<" ";

        }


        // As soon as numbers for each iteration are printed, we move to the

        // next row and give a line break otherwise all numbers

        // would get printed in 1 line.

        cout << endl;

    }
```

```
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern4(N);

    return 0;
}
```

Output

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

Pattern-5: Inverted Right Pyramid

Problem Statement: Given an integer **N**, print the following pattern :

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

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this pattern, we run the outer loop for N times as we have to print N rows and since we have to print a right-angled triangle/pyramid which must be inverted, the inner loop will run in decreasing order of stars, for eg: Row 1 (i=0) would contain N stars, Row 2 (i=1) would contain (N -1) stars and so on.

Code:

```
#include <bits/stdc++.h>
using namespace std;


void pattern5(int N)
{
    // This is the outer loop which will loop for the rows.
    for (int i = 0; i < N; i++)
    {
        // This is the inner loop which loops for the columns
       // no. of columns = (N - row index) for each line here.
        for (int j =N; j>i; j--)
        {
            cout <<"* ";
        }
```

```
        // As soon as stars for each iteration are printed, we move to the
        // next row and give a line break otherwise all stars
        // would get printed in 1 line.
        cout << endl;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern5(N);

    return 0;
}
```

Output

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

Pattern - 6: Inverted Numbered Right Pyramid

Problem Statement: Given an integer **N**, print the following pattern :

```
12345
1234
123
12
1
```

Here, N = 5.

Examples:

Input Format: N = 3

Result:

1 2 3

1 2

1

Input Format: N = 6

Result:

1 2 3 4 5 6

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this pattern, we run the outer loop for N times as we have to print N rows and since we have to print a right-angled triangle/pyramid which must be inverted, so the inner loop will run from 1 to (N-i)th integer in every row till we reach the Nth row where only '1' would be left to get printed. For eg: in the 1st-row numbers from 1 to N get printed, in the 2nd-row numbers from 1 to (N-1) get printed, and so on.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern6(int N)

{

    // This is the outer loop which will loop for the rows.

    for (int i = 0; i < N; i++)
```

```
    {
         // This is the inner loop which loops for the columns
        // no. of columns = (N - row index) for each line here
        // as we have to print an inverted pyramid.
        // (N-j) will give us the numbers in a row starting from 1 to N-i.
        for (int j =N; j>i; j--)
        {
            cout <<N-j+1<<" ";
        }

        // As soon as numbers for each iteration are printed, we move to the
        // next row and give a line break otherwise all numbers
        // would get printed in 1 line.
        cout << endl;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern6(N);

    return 0;
}
```

Output

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

Pattern - 7: Star Pyramid

Problem Statement: Given an integer **N**, print the following pattern :

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this particular pattern, we run the outer loop for N times as we have to print N rows as usual. Now, the question arises what will be the logic behind the inner loop?

As we can clearly observe that for each row there are some spaces that get printed then some stars and then again some spaces giving it a final pyramidal look. For eg: In the first row (i=0) there are 4 spaces, 1 star, then again 4 spaces. In the second row (i=1) there are 3 spaces, 3 stars, then again 3 spaces so we can say that there are N-i-1 spaces, 2*i+1 stars, and then again N-i-1 spaces for each row where i is the row index. We thus simply run 3 inner loops first for printing the spaces, then the stars, and then the spaces again.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern7(int N)

{

    // This is the outer loop which will loop for the rows.

    for (int i = 0; i < N; i++)

    {

        // For printing the spaces before stars in each row

        for (int j =0; j<N-i-1; j++)

        {

            cout <<" ";

        }


        // For printing the stars in each row

        for(int j=0;j< 2*i+1;j++){


            cout<<"*";

        }


        // For printing the spaces after the stars in each row

         for (int j =0; j<N-i-1; j++)

        {

            cout <<" ";

        }


        // As soon as the stars for each iteration are printed, we move to the

        // next row and give a line break otherwise all stars
```

```
        // would get printed in 1 line.
        cout << endl;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern7(N);

    return 0;
}
```

Output

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

Pattern - 8: Inverted Star Pyramid

Problem Statement: Given an integer **N**, print the following pattern :

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

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this particular pattern, we run the outer loop for N times as we have to print N rows as usual. Now, the question arises what will be the logic behind the inner loop?

Similar to the last pattern, we can clearly observe that for each row there are some spaces that get printed then some stars, and then again some spaces giving it an inverse pyramidal look. For eg: In the first row (i=0) there are 0 spaces, 9 stars, then again 0 spaces. In the second row (i=1) there is 1 space, 7 stars, then again 1 space so we can say that there are i spaces, 2*N - (2*i+1) stars, and then again i space for each row where i is the row index. We thus simply run 3 inner loops, first for printing the spaces, then the stars, and then the spaces again.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern8(int N)

{

    // This is the outer loop which will loop for the rows.

    for (int i = 0; i < N; i++)

    {
```

```
        // For printing the spaces before stars in each row
        for (int j =0; j<i; j++)
        {
            cout <<" ";
        }

        // For printing the stars in each row
        for(int j=0;j< 2*N -(2*i +1);j++){

            cout<<"*";
        }

        // For printing the spaces after the stars in each row
         for (int j =0; j<i; j++)
        {
            cout <<" ";
        }

        // As soon as the stars for each iteration are printed, we move to the
        // next row and give a line break otherwise all stars
        // would get printed in 1 line.
        cout << endl;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern8(N);

    return 0;
}
```

Output

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

Pattern - 9: Diamond Star Pattern

Problem Statement: Given an integer **N**, print the following pattern :

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

This pattern is just a mixture of the last two patterns (erect pyramid and inverted pyramid). Firstly, we will print the erect pyramid and then an inverted pyramid below it.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void erect_pyramid(int N)

{

    // This is the outer loop which will loop for the rows.

    for (int i = 0; i < N; i++)

    {

        // For printing the spaces before stars in each row

        for (int j =0; j<N-i-1; j++)

        {

            cout <<" ";

        }


        // For printing the stars in each row

        for(int j=0;j< 2*i+1;j++){
```

```
            cout<<"*";
        }

        // For printing the spaces after the stars in each row
         for (int j =0; j<N-i-1; j++)
        {
            cout <<" ";
        }

        // As soon as the stars for each iteration are printed, we move to the
        // next row and give a line break otherwise all stars
        // would get printed in 1 line.
        cout << endl;
    }
}

void inverted_pyramid(int N)
{
    // This is the outer loop which will loop for the rows.
    for (int i = 0; i < N; i++)
    {
        // For printing the spaces before stars in each row
        for (int j =0; j<i; j++)
        {
            cout <<" ";
        }

        // For printing the stars in each row
        for(int j=0;j< 2*N -(2*i +1);j++){

            cout<<"*";
        }

        // For printing the spaces after the stars in each row
         for (int j =0; j<i; j++)
```

```
        {
            cout <<" ";
        }

        // As soon as the stars for each iteration are printed, we move to the
        // next row and give a line break otherwise all stars
        // would get printed in 1 line.
        cout << endl;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    erect_pyramid(N);
    inverted_pyramid(N);

    return 0;
}
```

Output

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

Pattern - 10: Half Diamond Star Pattern

Problem Statement: Given an integer **N**, print the following pattern :

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

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the '*' inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this problem, we have to print only the right half of the star diamond pattern as discussed in the previous article. So, as we can observe from the examples for N = 3 we have 5 rows, and for N = 6 we have 11 rows, hence the outer loop will run for 2*N -1 times. For the inner loop where we print the stars if row no. is less than or equal to N, then we observe that the stars which are printed in each row are equal to the row index itself. But, when i becomes more than N, then the no. of stars decreases by 1 with each increasing row. So, therefore the stars printed would be 2*N - i after i becomes greater than N.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern10(int N)

{

      // Outer loop for number of rows.

      for(int i=1;i<=2*N-1;i++){


          // stars would be equal to the row no. uptill first half

          int stars = i;


          // for the second half of the rotated triangle.

          if(i>N) stars = 2*N-i;


          // for printing the stars in each row.

          for(int j=1;j<=stars;j++){

              cout<<"*";

          }
```

```
            // As soon as the stars for each iteration are printed, we move to
the
            // next row and give a line break otherwise all stars
            // would get printed in 1 line.
            cout<<endl;
        }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    pattern10(N);

    return 0;
}
```

Output

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

Pattern - 11: Binary Number Triangle Pattern

Problem Statement: Given an integer **N**, print the following pattern :

```
1
0 1
1 0 1
0 1 0 1
1 0 1 0 1
```

Here, N = 5.

Examples:

Input Format: N = 3

Result:

1

01

101

Input Format: N = 6

Result:

1

01

101

0101

10101

010101

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this problem, we have to print binary digits alternatively in each row and column as shown in the examples. Let's say that the first row starts with the binary digit '1', the second row must start with '0' and then the 3rd row with '1' again, and so on. Similar is the case for the columns as well. Initially, we declare a start variable and set it to 1 for the first row. For even no. of rows, the start variable is 1 and for odd it is 0. Now for the inner loop, the numbers are printed i times (i is the row index) alternatively by simply subtracting the start variable from 1 after each iteration.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern11(int N)

{
```

```
      // First row starts by printing a single 1.
      int start =1;

      // Outer loop for the no. of rows
      for(int i=0;i<N;i++){

          // if the row index is even then 1 is printed first
          // in that row.
          if(i%2 ==0) start = 1;

          // if odd, then the first 0 will be printed in that row.
          else start = 0;

          // We alternatively print 1's and 0's in each row by using
          // the inner for loop.
          for(int j=0;j<=i;j++){
              cout<<start;
              start = 1-start;
          }


        // As soon as the numbers for each iteration are printed, we move to
the
        // next row and give a line break otherwise all numbers
        // would get printed in 1 line.
        cout<<endl;
      }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    pattern11(N);
```

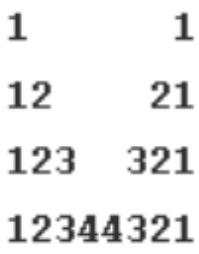
```
    return 0;
}
```

Output

```
1
01
101
0101
10101
```

Pattern - 12: Number Crown Pattern

Problem Statement: Given an integer **N**, print the following pattern :



Here, N = 5.

Examples:

Input Format: N = 3

Result:

```
1    1
12  21
123321
```

Input Format: N = 6

Result:

```
1          1
12        21
12       321
1234    4321
12345  54321
123456654321
```

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this problem, we want to print a combination of a numbered pyramid and a reverse-numbered pyramid. So, as per our observation in each row, numbers are printed from 1 to the row number and then some spaces and then again numbers from 1 to the row number but in reverse order. So, the outer loop will run from 1 to N and there will be three inner loops for numbers, spaces, and then again numbers.

The first inner loop will have numbers printed from 1 to the row number, the second will print the spaces (8 spaces in row 1, 6 spaces in row 2, and so on) and then the last loop will run from row number to 1 in decreasing manner. For spaces, we can say that initially, spaces are 2*(N-1) for Row 1 where N is the total no. of rows and then the spaces decrease by 2 in each iteration till the last row is reached.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern12(int N)

{

      // initial no. of spaces in row 1.

      int spaces = 2*(N-1);


      // Outer loop for the number of rows.

      for(int i=1;i<=N;i++){


          // for printing numbers in each row

          for(int j=1;j<=i;j++){

              cout<<j;

          }


          // for printing spaces in each row

          for(int j = 1;j<=spaces;j++){

              cout<<" ";
```

```
        }

        // for printing numbers in each row
        for(int j=i;j>=1;j--){
            cout<<j;
        }

        // As soon as the numbers for each iteration are printed, we move to
the
        // next row and give a line break otherwise all numbers
        // would get printed in 1 line.
        cout<<endl;

        // After each iteration nos. increase by 2, thus
        // spaces will decrement by 2.
        spaces-=2;
    }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    pattern12(N);

    return 0;
}
```

Output

```
1         1
12       21
123     321
1234   4321
1234554321
```

Pattern - 13: Increasing Number Triangle Pattern

Problem Statement: Given an integer **N**, print the following pattern :

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Here, N = 5.

Examples:

Input Format: N = 3

Result:

1

2 3

4 5 6

Input Format: N = 6

Result:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

16 17 18 19 20 21

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this problem, we just have to print the right-angled number pyramid but here, we also have to increase the number each time we print it. For printing, the right-angled pyramid as we know the outer loop runs for N times and the inner loop runs for i times. Now, to print an increasing number pyramid we just

have to increment the number inside the inner loop so that after printing the number each time it increases by 1.

Code:

```
#include <bits/stdc++.h>
using namespace std;

void pattern13(int N)
{
      // starting number
      int num =1;

      // Outer loop for the number of rows.
      for(int i=1;i<=N;i++){

          // Inner loop will loop for i times and
          // print numbers increasing by 1 each time.
          for(int j=1;j<=i;j++){
              cout<<num<<" ";
              num =num+1;
          }
          // As soon as the numbers for each iteration are printed, we move to
the
          // next row and give a line break otherwise all numbers
          // would get printed in 1 line.
          cout<<endl;

      }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    pattern13(N);

    return 0;
```

```
}
```

Output

```
1
2  3
4  5  6
7  8  9  10
11  12  13  14  15
```

Pattern-14: Increasing Letter Triangle Pattern

Problem Statement: Given an integer **N**, print the following pattern :

A

AB

ABC

ABCD

ABCDE

Here, N = 5.

Examples:

Input Format: N = 3

Result:

A

A B

A B C

Input Format: N = 6

Result:

A

A B

A B C

A B C D

A B C D E

A B C D E F

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns.

In this pattern problem, instead of numbers, we have to print alphabets hence making the pattern look like a right-angled triangle. So, the outer loop will run for N rows and the inner loop will loop for i alphabets in each row where i is the row number. Alphabets in each row will start from A each time we enter a new row and will loop till (A+i)th alphabet in that row.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern14(int N)

{


      // Outer loop for the number of rows.

      for(int i=0;i<N;i++){


          // Inner loop will loop for i times and

          // print alphabets from A to A + i.

          for(char ch = 'A'; ch<='A'+i;ch++){

              cout<<ch<<" ";


          }

          // As soon as the letters for each iteration are printed, we move to
the

          // next row and give a line break otherwise all letters

          // would get printed in 1 line.

          cout<<endl;


      }

}


int main()
```

```
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    pattern14(N);

    return 0;
}
```

Output

```
A
A B
A B C
A B C D
A B C D E
```

Pattern-15: Reverse Letter Triangle Pattern

Problem Statement: Given an integer **N,** print the following pattern :

ABCDE

ABCD

ABC

AB

A

Here, N = 5.

Examples:

Input Format: N = 3

Result:

A B C

A B

A

Input Format: N = 6

Result:

A B C D E F

A B C D E

A B C D

A B C

A B

A

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

This pattern problem is very similar to the last pattern problem we did where we had to print an increasing letter pyramid pattern but this time we have to print it in the reverse fashion. So, the outer loop will run for N rows and the inner loop will loop for N-i-1 alphabets in each row (where i is the row number) because the 1st row will have alphabets from A to A+(N-1). Alphabets in each row will start from A each time we enter a new row and will loop till (A+N-i-1)th alphabet in that row. In this way, the last row will only contain the alphabet A at last.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern15(int N)

{


      // Outer loop for the number of rows.

      for(int i=0;i<N;i++){


          // Inner loop will loop for i times and

          // print alphabets from A to A + (N-i-1).

          for(char ch = 'A'; ch<='A'+(N-i-1);ch++){

              cout<<ch<<" ";


          }
```

```
            // As soon as the letters for each iteration are printed, we move to
the
            // next row and give a line break otherwise all letters
            // would get printed in 1 line.
            cout<<endl;

        }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    pattern15(N);

    return 0;
}
```

Output

```
A B C D E
A B C D
A B C
A B
A
```

Pattern - 16: Alpha-Ramp Pattern

Problem Statement: Given an integer **N**, print the following pattern :

A

BB

CCC

DDDD

EEEEE

Here, N = 5.

Examples:

Input Format: N = 3

Result:

A

B B

C C C

Input Format: N = 6

Result:

A

B B

C C C

D D D D

E E E E E

F F F F F F

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this pattern problem, as we can observe we have to print a right-angled pyramid just like our last pattern but with a twist. Here, in every row, we have to print the same character i times where i is the row number. For example, the 1st row will print 1 A, the 2nd row will print 2 B's, and so on. So, similar to the previous patterns the outer loop will loop for N times and the inner loop for i times with the character value incrementing each time we enter a new row.

Code:

```
#include <bits/stdc++.h>

using namespace std;


void pattern16(int N)

{
```

```
      // Outer loop for the number of rows.
      for(int i=0;i<N;i++){

          // Defining character for each row.
          char ch = 'A'+i;
          for(int j=0;j<=i;j++){

              // same char is to be printed i times in that row.
              cout<<ch<<" ";

          }
          // As soon as the letters for each iteration are printed, we move to
the
          // next row and give a line break otherwise all letters
          // would get printed in 1 line.
          cout<<endl;

      }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
    pattern16(N);

    return 0;
}
```

Output

```
A
B B
C C C
D D D D
E E E E E
```

Pattern - 17: Alpha-Hill Pattern

Problem Statement: Given an integer **N**, print the following pattern :

Here, N = 4.

Examples:

Input Format: N = 3

Result:

```
  A
 ABA
ABCBA
```

Input Format: N = 6

Result:

```
     A
    ABA
   ABCBA
  ABCDCBA
 ABCDEDCBA
ABCDEFEDCBA
```

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.

- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns.

In this problem, we have to print an alphabet triangle as shown in the examples above where we can clearly observe that the first row has only the letter A in the middle and some spaces on either side of it and the second row has the letters ABA and some spaces again on both sides. So, we observe from this that first there are some spaces, then letters increasing from A to A + i where i is the row number and then decreasing back to A, then finally some more spaces to the end. Hence, like all the previous patterns the outer loop will loop for N times and there will be three inner loops in this pattern.

The first inner loop is for printing the (N-i-1) spaces (1st row -> 4 spaces, 2nd row -> 3 spaces, and so on), the second one for printing the characters, and then the last one again for printing the spaces. For the character loop, we define the breakpoint till where the character must increase and after that, it must decrease. The breakpoint can be defined by (2*i+1)/2 for each row.

Code:

C++Java

```
#include <bits/stdc++.h>

using namespace std;


void pattern17(int N)

{


      // Outer loop for the number of rows.

      for(int i=0;i<N;i++){


          // for printing the spaces.

          for(int j=0;j<N-i-1;j++){

              cout<<" ";

          }


          // for printing the characters.

          char ch = 'A';

          int breakpoint = (2*i+1)/2;

          for(int j=1;j<=2*i+1;j++){


              cout<<ch;

              if(j <= breakpoint) ch++;

              else ch--;

          }
```

```
            // for printing the spaces again after characters.
            for(int j=0;j<N-i-1;j++){
                cout<<" ";
            }

            // As soon as the letters for each iteration are printed, we move to
the
            // next row and give a line break otherwise all letters
            // would get printed in 1 line.
            cout<<endl;

        }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern17(N);

    return 0;
}
```

Output

```
    A
   ABA
  ABCBA
 ABCDCBA
ABCDEDCBA
```

Pattern-18: Alpha-Triangle Pattern

Problem Statement: Given an integer **N**, print the following pattern :

```
E
D E
C D E
B C D E
A B C D E
```

Here, N = 5.

Examples:

Input Format: N = 3

Result:

C

B C

A B C

Input Format: N = 6

Result:

F

E F

D E F

C D E F

B C D E F

A B C D E F

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns.

In this problem, we have to print an alpha triangle as shown in the examples above. We observe from the examples that each row ends with the letter E in the case when N = 5 (‘A’ + 4). Also the triangle has to be right-angled so like the previous patterns, the outer loop will run for N times and the inner loop for i times. In the inner loop, we’ll start from the letter that comes i before the (‘A’ + N -1)th letter and then run the loop till we reach (‘A’ + N-1) in every row. For example, for N = 5 in each row, the letters will be printed from ‘E’ - i to ‘E’ where i is the row index.

Code:

C++Java

```
#include <bits/stdc++.h>
using namespace std;


void pattern18(int N)
{
      // Outer loop for the no. of rows.
      for(int i=0;i<N;i++){

          // Inner loop for printing the alphabets from
          // A + N -1 -i (i is row no.) to A + N -1 ( E in this case).
          for(char ch =('A'+N-1)-i;ch<=('A'+N-1);ch++){

              cout<<ch<<" ";
          }

          // As soon as the letters for each iteration are printed, we move to
the
          // next row and give a line break otherwise all letters
          // would get printed in 1 line.
          cout<<endl;
      }
}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
```

```
    pattern18(N);


    return 0;
}
```

Output

```
E
D E
C D E
B C D E
A B C D E
```

Pattern-19: Symmetric-Void Pattern

Problem Statement: Given an integer **N**, print the following pattern :

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

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns.

Contrary to the previous patterns, this pattern observes symmetry. We can clearly see that the first half of the pattern is just a mirror image of the second half of the pattern. If we observe the first part, we see that like some previous patterns, it can also be divided into 3 parts i.e. stars, spaces, and then stars. In the first row, there are no spaces and 10 stars, in the second row, there are 2 spaces and 8 stars, and so on. So, we initialize the spaces with 0 initially which will eventually increment by 2 whenever we enter a new row. The stars, however, will be twice the row number, half of the stars would be printed before the spaces and half after the spaces.

Similar will be the case of the second half of the pattern too. The initial number of spaces would be equal to 2*(N-1) and 1 star each would be printed on either side of the spaces initially. The spaces will decrease by 2 and the stars will increase by 2 in each row.

Code:

C++Java

```
#include <bits/stdc++.h>

using namespace std;
```

```
void pattern19(int N)
{
      // for the upper half of the pattern.

      // initial spaces.
      int iniS = 0;
      for(int i=0;i< N;i++){

          //for printing the stars in the row.
          for(int j=1;j<=N-i;j++){
              cout<<"*";
          }

          //for printing the spaces in the row.
          for(int j=0;j<iniS;j++){
              cout<<" ";
          }

          //for printing the stars in the row.
          for(int j=1;j<=N-i;j++){
              cout<<"*";
          }

          // The spaces increase by 2 every time we hit a new row.
          iniS+=2;

          // As soon as the stars for each iteration are printed, we move to
the
          // next row and give a line break otherwise all stars
          // would get printed in 1 line.
          cout<<endl;
      }

      // for lower half of the pattern

      // initial spaces.
```

```
      iniS = 2*N -2;
      for(int i=1;i<=N;i++){

          //for printing the stars in the row.
          for(int j=1;j<=i;j++){
              cout<<"*";
          }

          //for printing the spaces in the row.
          for(int j=0;j<iniS;j++){
              cout<<" ";
          }

          //for printing the stars in the row.
          for(int j=1;j<=i;j++){
              cout<<"*";
          }

          // The spaces decrease by 2 every time we hit a new row.
          iniS-=2;

          // As soon as the stars for each iteration are printed, we move to
the
          // next row and give a line break otherwise all stars
          // would get printed in 1 line.
          cout<<endl;
      }

}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;
```

```
    pattern19(N);


    return 0;
}
```

Output

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

Pattern - 20: Symmetric-Butterfly Pattern

Problem Statement: Given an integer **N,** print the following pattern :

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

Here, N = 5.

Examples:

Input Format: N = 3

Result:

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

Input Format: N = 6

Result:

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

In this problem, we have to print a butterfly-like star pattern. This pattern is very similar to Pattern 10 in this series as here we can see that for Row 1 we have 2 stars, and 8 spaces, and for Row 2 we have 4 stars and 6 spaces, and so on. Also, after the nth row, the stars decrease and the spaces increase. So, the outer loop will run for 2*n -1 times (n times when spaces > stars and then n-1 when stars > spaces). There will be 3 inner loops, one to print stars, then spaces, and then again stars. Spaces will decrement by 2 as i increases and when i reaches n, then spaces decrement by 2 every time we enter a new row. When i<n, the stars printed in each row are 2*i, and as soon as i>n, the stars in each row would be twice of 2*n-i.

Code:

C++Java

```
#include <bits/stdc++.h>

using namespace std;
```

```
void pattern20(int n)
{
      // initialising the spaces.
      int spaces = 2*n-2;

      // Outer loop for printing row.
      for(int i = 1;i<=2*n-1;i++){

          // stars for first half
          int stars = i;

          // stars for the second half.
          if(i>n) stars = 2*n - i;

          //for printing the stars
          for(int j=1;j<=stars;j++){
              cout<<"*";
          }

          //for printing the spaces
          for(int j = 1;j<=spaces;j++){
              cout<<" ";
          }

          //for printing the stars
          for(int j = 1;j<=stars;j++){
              cout<<"*";
          }

          // As soon as the stars for each iteration are printed, we move to
the
          // next row and give a line break otherwise all stars
          // would get printed in 1 line.
          cout<<endl;
          if(i<n) spaces -=2;
          else spaces +=2;
```

```
        }

}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern20(N);

    return 0;
}
```

Output

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

Pattern - 21: Hollow Rectangle Pattern

Problem Statement: Given an integer **N**, print the following pattern :

Here, N = 5.

Examples:

Input Format: N = 3

Result:

```
***
```

```
* *
***
```

Input Format: N = 6

Result:

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

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns.

This problem is a very simple one where we just have to print a hollow rectangle. So, after observation, we see that the stars are only printed at the border of the rectangle, and on the rest of the positions, spaces are printed. Hence, the outer loop and inner loop both will loop for n times and in the inner loop we add a condition that if the row index or column index is equal to 0 or n-1, then stars should get printed otherwise spaces should be printed.

Code:

C++Java

```
#include <bits/stdc++.h>
using namespace std;


void pattern21(int n)
{
     // outer loop for no. of rows.
     for(int i=0;i<n;i++){
```

```
        // inner loop for printing the stars at borders only.
        for(int j=0;j<n;j++){

            if(i==0 || j==0 || i==n-1 || j==n-1)
               cout<<"*";

            // if not border index, print space.
            else cout<<" ";
        }

         // As soon as the stars for each iteration are printed, we move to
the
         // next row and give a line break otherwise all stars
         // would get printed in 1 line.
         cout<<endl;
     }

}

int main()
{
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern21(N);

    return 0;
}
```

Output

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

Pattern - 22: The Number Pattern

Problem Statement: Given an integer **N**, print the following pattern :

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

Here, N = 4.

Examples:

Input Format: N = 3

Result:

```
3 3 3 3 3
3 2 2 2 3
3 2 1 2 3
3 2 2 2 3
3 3 3 3 3
```

Input Format: N = 6

Result:

```
6 6 6 6 6 6 6 6 6 6 6
6 5 5 5 5 5 5 5 5 5 6
6 5 4 4 4 4 4 4 4 5 6
6 5 4 3 3 3 3 3 4 5 6
6 5 4 3 2 2 2 3 4 5 6
6 5 4 3 2 1 2 3 4 5 6
6 5 4 3 2 2 2 3 4 5 6
6 5 4 3 3 3 3 3 4 5 6
6 5 4 4 4 4 4 4 4 5 6
6 5 5 5 5 5 5 5 5 5 6
6 6 6 6 6 6 6 6 6 6 6
```

Solution

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

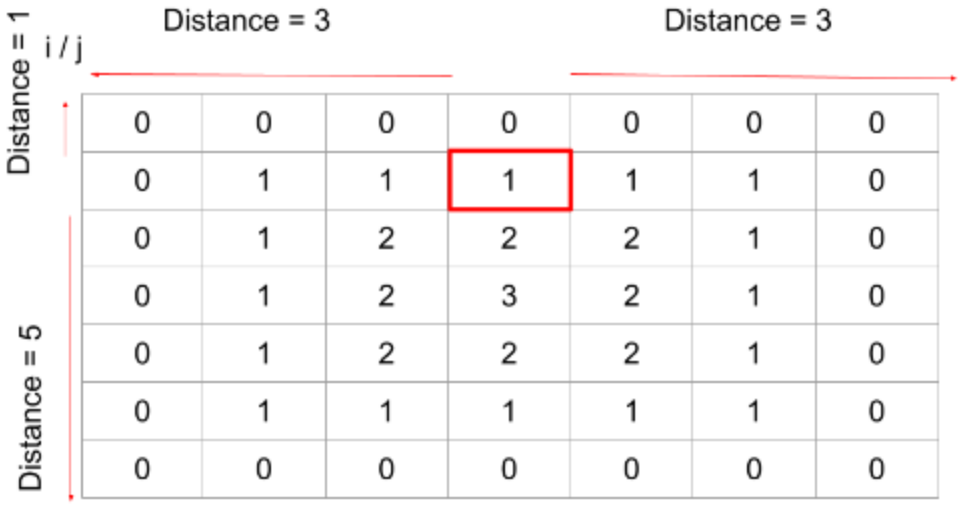
Problem Link

Approach:

There are 4 general rules for solving a pattern-based question :

- We always use nested loops for printing the patterns. For the outer loop, we count the number of lines/rows and loop for them.
- Next, for the inner loop, we focus on the number of columns and somehow connect them to the rows by forming a logic such that for each row we get the required number of columns to be printed.
- We print the numbers inside the inner loop.
- Observe symmetry in the pattern or check if a pattern is a combination of two or more similar patterns or not.

This problem is not generally asked in the interviews but it is good to practice such problems for the sake of logic building. So, what we can observe from the above examples is that on the perimeter of the square, there is an integer no. N which decreases by 1 as we move inside the square level-wise. Since this cannot be printed directly, we print it in reverse fashion (0's at the border of the square, then 1 in the inner perimeter, then 2, and so on) and then subtract the whole pattern by N at the end which just makes the outermost perimeter filled with the number N, inner perimeter with N-1 and finally the centermost element with the integer 1. The outer and the inner loop will run for the same number of times (since we have to print square) i.e, 2*N-1 times and the inner loop would print the numbers based on the logic as described below (for N = 4):



As per the above illustration, we take the minimum distance of the current cell from each of the ends of the square (min(1,3,3,5)) and make the current cell value equal to that number.

- The distance of the current cell from the top can be written as the row number i.e, **i.**
- The distance of the current cell from the left can be written as the column number i.e, **j.**
- The distance of the current cell from the right can be written as **(2*n-2) - j.**
- The distance of the current cell from the bottom can be written as **(2*n-2) - i.**

Code:

C++Java

```
#include <bits/stdc++.h>
using namespace std;

void pattern22(int n)
{
     // Outer loop for no. of rows
     for(int i=0;i<2*n-1;i++){

          // inner loop for no. of columns.
          for(int j=0;j<2*n-1;j++){

               // Initialising the top, down, left and right indices of a cell.
               int top = i;
               int bottom = j;
               int right = (2*n - 2) - j;
               int left = (2*n - 2) - i;

               // Min of 4 directions and then we subtract from n
               // because previously we would get a pattern whose border
               // has 0's, but we want with border N's and then decreasing
inside.
               cout<<(n- min(min(top,bottom), min(left,right)))<<" ";
          }

          // As soon as the numbers for each iteration are printed, we move to
the
          // next row and give a line break otherwise all numbers
          // would get printed in 1 line.
          cout<<endl;
     }

}

int main()
{
```

```
    // Here, we have taken the value of N as 5.
    // We can also take input from the user.
    int N = 5;

    pattern22(N);

    return 0;
}
```

Output

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