**6.Spiral matrix**

Problem Statement:

Create an n × n square matrix dynamically taking n as an input from the user.

Fill the matrix with integers 1, 2, ..., n in a spiral order starting from the left top corner of the matrix. Print the matrix.

• Input example :

1. 5
2. 9

• Output example :

1. 1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

1. 1 2 3 4 5 6 7 8 9

32 33 34 35 36 37 38 39 10

31 56 57 58 59 60 61 40 11

30 55 72 73 74 75 62 41 12

29 54 71 80 81 76 63 42 13

28 53 70 79 78 77 64 43 14

27 52 69 68 67 66 65 44 15

26 51 50 49 48 47 46 45 16

25 24 23 22 21 20 19 18 17

Proposed C Code:

/\* ------- main.c ------- \*/

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int n, c = 1;

    printf("Enter N: ");

    scanf("%d", &n);

    int \*\*arr = (int \*\*)malloc(n \* sizeof(int)); *// initializing a 2D array*

    for (int i = 0; i < n; i++)

    {

        arr[i] = (int \*)malloc(n \* sizeof(int));

    }

    int l = 0, r = n - 1, t = 0, b = n - 1; *// taking 4 pointers(left,right,top,bottom)*

    while (l <= r || t <= b)

    {

        for (int i = l; i <= r; i++) *// moving left to right*

        {

            arr[t][i] = c;

            c++;

        }

        t++; *// incrementing top*

        for (int i = t; i <= b; i++) *// moving top to bottom*

        {

            arr[i][r] = c;

            c++;

        }

        r--; *// decrementing right*

        for (int i = r; i >= l; i--) *// moving right to left*

        {

            arr[b][i] = c;

            c++;

        }

        b--; *// decrementing bottom*

        for (int i = b; i >= t; i--) *// moving bottom to top*

        {

            arr[i][l] = c;

            c++;

        }

        l++; *// incrementing left*

    }

    for (int i = 0; i < n; i++) *// printing the array*

    {

        for (int j = 0; j < n; j++)

        {

            printf(" %3d ", arr[i][j]);

        }

        printf("\n");

    }

    return 0;

}

/\* ---------------------- \*/

Conclusion:

The proposed algorithm has a runtime of O(n2), where n is the number of rows or columns of the matrix.

Limitations and assumptions for this algorithm include:

1. The time complexity of this program is of the order 2 rather than linear.
2. Here we are assuming that the number of rows and number of columns are same for the matrix.