CMPE150.05

23.12.2018

CMPE150 ASSIGNMENT 3

1. I chose option 2 (with for/whiles).
2. The program handles all labels (F, R, C, D and N)
3. My code is as follows:

```
package ECK2018400018;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Arrays;
import java.util.Scanner;

public class ECK2018400018
{
	public static void main(String[] args) throws FileNotFoundException
	{
		Scanner scf = new Scanner(new File("input.txt"));
		String dimensions = scf.nextLine();
		int separator = dimensions.indexOf("x");
		String heights = dimensions.substring(0, separator);
		String widths = dimensions.substring(separator+1);
		int height = Integer.parseInt(heights);
		int width = Integer.parseInt(widths);

		int inta[][] = new int[height][width];
		char cha[][] = new char[height][width];

		for (int i = 0; i < height; i++)
		{
			for (int j = 0; j < width; j++)
			{
				String bundle = scf.next();
				cha[i][j] = bundle.charAt(0);
				inta[i][j] = Integer.parseInt(bundle.substring(1));
			}
		}

		printArray(inta);

		for (int i = 0; i < height; i++)
		{
			for (int j = 0; j < width; j++)
			{
				if (cha[i][j] == 'R')
				{
					int rmax = Integer.MIN_VALUE;
```

```
                        for (int k = 0; k < width; k++)
                        {
                                if (inta[i][k] > rmax)
                                {
                                        rmax = inta[i][k];
                                }
                        }

                        inta[i][j] = rmax;
                }

                else if (cha[i][j] == 'C')
                {
                        int c[] = new int[height];

                        for (int k = 0; k < height; k++)
                        {
                                c[k] = inta[k][j];
                        }

                        Arrays.sort(c);

                        int med = 0;
                        if (c.length % 2 == 0)
                        {
                                med = c[(c.length/2)-1];
                        }
                        else
                        {
                                med = c[c.length/2];
                        }

                        inta[i][j] = med;
                }

                else if (cha[i][j] == 'D')
                {
                        int dsum = 0;
                        int count = 0;

                        // sum of diagonals towards lower-right, including
the number itself
                        int r_d = Math.min(width-j-1, height-i-1);
                        for (int k = 0; k <= r_d; k++)
                        {
                                dsum += inta[i+k][j+k];
                                count++;
                        }

                        // sum of diagonals towards upper-right
                        int r_u = Math.min(width-j-1, i);
                        for (int k = 1; k <= r_u; k++)
                        {
                                dsum += inta[i-k][j+k];
                                count++;
```

```
                                }

                                // sum of diagonals towards lower-left
                                int l_d = Math.min(j, height-i-1);
                                for (int k = 1; k <= l_d; k++)
                                {
                                        dsum += inta[i+k][j-k];
                                        count++;
                                }

                                // sum of diagonals towards upper-left
                                int l_u = Math.min(j, i);
                                for (int k = 1; k <= l_u; k++)
                                {
                                        dsum += inta[i-k][j-k];
                                        count++;
                                }

                                inta[i][j] = dsum / count;
                        }

                        else if (cha[i][j] == 'N')
                        {
                                checkNeighbors(inta, cha, i, j);
                        }
                }
        }
        printArray(inta);
        scf.close();
    }

    // simple method for printing out every element of a 2D array like a
rectangular matrix
    public static void printArray(int inta[][])
    {
        int height = inta.length;
        int width = inta[0].length;
        for (int i = 0; i < height; i++)
        {
                for (int j = 0; j < width; j++)
                {
                        System.out.print(inta[i][j] + " ");
                }
                System.out.println();
        }
        System.out.println();
    }

    // a recursive method for checking the four neighbors of cells labeled "N",
replaces processed "N" cells with "n"
    public static void checkNeighbors(int inta[][], char cha[][], int i, int j)
    {
        if (j+1 <= cha[0].length-1)
        {
                if (cha[i][j+1] == 'N')
```

```
                {
                        inta[i][j+1] = inta[i][j];
                        cha[i][j] = 'n';
                        if (j+1 <= cha[0].length-1)
                        {
                                checkNeighbors(inta, cha, i, j+1);
                        }
                }
        }
        if (i+1 <= cha.length-1)
        {
                if (cha[i+1][j] == 'N')
                {
                        inta[i+1][j] = inta[i][j];
                        cha[i][j] = 'n';
                        if (i+1 <= cha.length-1)
                        {
                                checkNeighbors(inta, cha, i+1, j);
                        }
                }
        }
        if (j-1 >= 0)
        {
                if (cha[i][j-1] == 'N')
                {
                        inta[i][j-1] = inta[i][j];
                        cha[i][j] = 'n';
                        if (j-1 >= 0)
                        {
                                checkNeighbors(inta, cha, i, j-1);
                        }
                }
        }
        if (i-1 >= 0)
        {
                if (cha[i-1][j] == 'N')
                {
                        inta[i-1][j] = inta[i][j];
                        cha[i][j] = 'n';
                        if (i-1 >= 0)
                        {
                                checkNeighbors(inta, cha, i-1, j);
                        }
                }
        }
    }
}
```

4. Outputs of the program: