

## 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;

```

the number itself

```

// sum of diagonals towards lower-right, including

```

```

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:

```
ECK2018400018.java  input.txt x
1 6x4
2 D9 F8 N0 D7
3 N5 R3 R6 C7
4 C3 F2 N1 N2
5 F7 R1 C4 N3
6 R8 F8 N4 N0
7 N3 N5 D8 D9|

Problems Javadoc Declaration Console x
<terminated> ECK2018400018 (3) [Java Application] D:\Programs
9 8 0 7
5 3 6 7
3 2 1 2
7 1 4 3
8 8 4 0
3 5 8 9

4 8 0 5
5 7 7 3
4 2 1 1
7 7 1 1
8 8 1 1
3 3 6 5
```

```
ECK2018400018.java  input.txt x
1 14x7
2 F9 N9 F7 N7 N7 C0 C4
3 D0 R7 F6 R8 C4 R0 R2
4 N4 R4 C9 C5 D9 R4 D5
5 N6 N1 D3 R6 F8 R3 C7
6 R9 N4 N0 D3 C3 N6 D3
7 R7 N6 N6 R0 C5 N9 C5
8 C3 N4 R7 D3 R2 D2 C2
9 N6 F4 N5 D1 R8 D9 C2
10 R2 C6 F2 R7 R9 R4 F6
11 N7 F5 F5 N9 R4 N2 D2
12 C1 R8 C8 R5 D7 F1 R0
13 F9 R1 F0 N4 D0 N9 N0
14 C9 C4 C8 N3 N3 N9 R2
15 D3 F2 N4 N4 N2 R2 D7

Problems Javadoc Declaration Console x
<terminated> ECK2018400018 (3) [Java Application] D:\Programs
p 9 7 7 7 0 4
0 7 6 8 4 0 2
4 4 9 5 9 4 5
6 1 3 6 8 3 7
9 4 0 3 3 6 3
7 6 6 0 5 9 5
3 4 7 3 2 2 2
6 4 5 1 8 9 2
2 6 2 7 9 4 6
7 5 5 9 4 2 2
1 8 8 5 7 1 0
9 1 0 4 0 9 0
9 4 8 3 3 9 2
3 2 4 4 2 2 7

9 9 7 7 7 3 2
3 8 6 8 4 8 8
4 9 5 4 5 9 4
4 4 4 8 8 8 2
9 4 4 5 4 6 4
7 4 4 7 4 6 2
4 4 7 4 7 5 2
6 4 5 4 9 6 2
9 4 2 9 9 9 6
7 5 5 9 9 2 4
6 8 5 8 4 1 8
9 9 0 4 4 4 4
6 4 5 4 4 4 6
5 2 4 4 4 7 5
```