# Assignment 6

Introduction to programming in C

## Question 1

Given an n×n integer Matrix A and an positive number $\ell$ such that $2\ell + 1 \leq n$, print the $\ell$ window smoothing of A.

To get the $\ell$-window smoothing of A , we replace $A[i][j]$ with the sum of the values of the $2\ell + 1 \times 2\ell + 1$ submatrix of A with centre at $A[i][j]$ .

More precisely, the smoothed matrix

$$B[i,j] = \sum_{u=il}^{ih} \sum_{v=jl}^{jh} A[u][v]$$

where $il = \max(i - \ell, 0), \quad ih = \min(i + \ell, n - 1), \quad jl = \max(j - \ell, 0), \quad jh = \min(j + \ell, n - 1)$.

Input

The first line contains the dimension of the matrix n. Assume $n < 100$. The second line contains the smoothing parameter $\ell$. The next n lines contains the contents of the matrix A, each row per line.

Output

The smoothed matrix of A.

### Solution

```
#include <stdio.h>

int max(int a, int b){
    if(a>b)
        return a;
    return b;
}

int min(int a, int b){
    if(a<b)
        return a;
    return b;
}
```

```
int main() {
    int A[100][100];
    int B[100][100];

    int n,l,sum;

    scanf("%d",&n);
    scanf("%d",&l);

    for(int i = 0; i< n;i++){
        for(int j = 0; j< n;j++)
            scanf("%d",&A[i][j]);
    }

    for(int i = 0; i< n;i++){
        for(int j = 0; j< n;j++){

            int ih,il,jh,jl;
            sum =0;

            il = max(i-l,0);
            ih = min(i+l,n-1);
            jl = max(j-l,0);
            jh = min(j+l,n-1);


            for(int a=il;a<=ih;a++)
             for(int b=jl;b<=jh;b++)
                sum+=A[a][b];

            B[i][j] = sum;
        }
    }

    for(int i = 0; i< n;i++){
        for(int j = 0; j< n;j++){
            printf("%d ",B[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

# Question 2

**Simple Path Finding**

Given an n×n binary Matrix A , where each entry is 0 or 1. A has a unique path of 1's from A[0][0] to A[n-1][n-1]. The path always goes Right (R) or Down (D).

Write a C Program.to print the directions of this path.

Note: You can assume that there is exactly one correct path. All 1's in A are in this unique path, there are no dead ends.

Input

The first line contains the dimension of the matrix n. Assume n < 100. The second line contains the contents of the matrix A, each row per line.

Output

The path of 1's in the Matrix.

## Solution

```c
#include <stdio.h>

void findPath(int matrix[][100], int n, int x, int y, char* path, int pathIndex) {

    // If the destination is reached, print the path and return
    if (x == n - 1 && y == n - 1) {
        path[pathIndex] = '\0'; // Null terminate the string
        printf("%s\n", path);
        return;
    }

    // Move Right
    if (y + 1 < n && matrix[x][y + 1] == 1) {
        path[pathIndex] = 'R';
        findPath(matrix, n, x, y + 1, path, pathIndex + 1);
    }

    // Move Down
    if (x + 1 < n && matrix[x + 1][y] == 1) {
        path[pathIndex] = 'D';
        findPath(matrix, n, x + 1, y, path, pathIndex + 1);
    }
}

int main() {
    int n;
    scanf("%d", &n);

    int matrix[100][100];
    char path[200]; // Assuming the path will not be longer than 2n-1 steps

    // Read the matrix
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    findPath(matrix, n, 0, 0, path, 0);

    return 0;
}
```

# Question 3

**Recursive Path Finding**

Given an n×n binary Matrix A , where each entry is 0 or 1. A has a unique path of 1's from A[0][0] to A[n-1][n-1]. The path can go Right (R) Left (R) Down (D) or Up (U).

Write a C Program.to print the directions of this path.

Note: You can assume that there is exactly one correct path. All 1's in A need not be in this unique path, there can be dead ends.

Input

The first line contains the dimension of the matrix n. Assume n < 100. The second line contains the contents of the matrix A, each row per line.

Output

The path of 1's in the Matrix.

## Solution

```c
#include <stdio.h>

int findPath(int matrix[100][100], int n, int x, int y, char* path,
     int pathIndex) {

    // If the destination is reached, print the path and return
    if (x == n - 1 && y == n - 1) {
        path[pathIndex] = '\0'; // Null terminate the string
        printf("%s\n", path);
        return 1;
    }

    int last = 'I';

    if(pathIndex !=0)
     last = path[pathIndex - 1];


    // Try moving Right
    if (last != 'L' && y + 1 < n && matrix[x][y + 1] == 1) {
        path[pathIndex] = 'R';
        if(findPath(matrix, n, x, y + 1, path, pathIndex + 1))
            return 1;
    }

    // Try moving Left
    if (last != 'R' && y - 1 >= 0 && matrix[x][y - 1] == 1) {
        path[pathIndex] = 'L';
        if(findPath(matrix, n, x, y - 1, path, pathIndex + 1))
            return 1;
    }

    // Try moving Down
    if (last != 'U' && x + 1 < n && matrix[x + 1][y] == 1) {
        path[pathIndex] = 'D';
        if(findPath(matrix, n, x + 1, y, path, pathIndex + 1))
            return 1;
    }

    // Try moving Up
    if (last != 'D' && x - 1 >= 0 && matrix[x - 1][y] == 1) {
        path[pathIndex] = 'U';
        if(findPath(matrix, n, x - 1, y, path, pathIndex + 1))
            return 1;
    }

    return 0;
}

int main() {
    int n;
    scanf("%d", &n);

    int matrix[100][100];
    char path[1000];

    // Read the matrix
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    findPath(matrix, n, 0, 0, path, 0);

    return 0;
}
```