

Assignment 6

Introduction to programming in C

Question 1

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

To get the ℓ -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 $il = \max(i - \ell, 0)$, $ih = \min(i + \ell, n - 1)$, $jl = \max(j - \ell, 0)$, $jh = \min(j + \ell, n - 1)$.

Input

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

Output

The smoothed matrix of A.

Solution

```
1 #include <stdio.h>
2
3 int max(int a, int b){
4     if(a>b)
5         return a;
6     return b;
7 }
8
9 int min(int a, int b){
10     if(a<b)
11         return a;
12     return b;
13 }
```

```

14
15 int main() {
16     int A[100][100];
17     int B[100][100];
18
19     int n,l,sum;
20
21     scanf("%d",&n);
22     scanf("%d",&l);
23
24     for(int i = 0; i < n; i++){
25         for(int j = 0; j < n; j++){
26             scanf("%d",&A[i][j]);
27         }
28
29         for(int i = 0; i < n; i++){
30             for(int j = 0; j < n; j++){
31
32                 int ih,il,jh,jl;
33                 sum =0;
34
35                 il = max(i-1,0);
36                 ih = min(i+1,n-1);
37                 jl = max(j-1,0);
38                 jh = min(j+1,n-1);
39
40
41                 for(int a=il; a<=ih; a++){
42                     for(int b=jl; b<=jh; b++){
43                         sum+=A[a][b];
44
45                     B[i][j] = sum;
46                 }
47             }
48
49             for(int i = 0; i < n; i++){
50                 for(int j = 0; j < n; j++){
51                     printf("%d ",B[i][j]);
52                 }
53                 printf("\n");
54             }
55             return 0;
56 }

```

Question 2

Simple Path Finding

Given an $n \times n$ 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

```
1 #include <stdio.h>
2
3 void findPath(int matrix[][100], int n, int x, int y, char* path,
4             int pathIndex) {
5     // If the destination is reached, print the path and return
6     if (x == n - 1 && y == n - 1) {
7         path[pathIndex] = '\0'; // Null terminate the string
8         printf("%s\n", path);
9         return;
10    }
11
12    // Move Right
13    if (y + 1 < n && matrix[x][y + 1] == 1) {
14        path[pathIndex] = 'R';
15        findPath(matrix, n, x, y + 1, path, pathIndex + 1);
16    }
17
18    // Move Down
19    if (x + 1 < n && matrix[x + 1][y] == 1) {
20        path[pathIndex] = 'D';
21        findPath(matrix, n, x + 1, y, path, pathIndex + 1);
22    }
23 }
24
25 int main() {
26     int n;
27     scanf("%d", &n);
28
29     int matrix[100][100];
30     char path[200]; // Assuming the path will not be longer than 2n
31                     // -1 steps
32
33     // Read the matrix
34     for (int i = 0; i < n; i++) {
35         for (int j = 0; j < n; j++) {
36             scanf("%d", &matrix[i][j]);
37         }
38     }
39
40     findPath(matrix, n, 0, 0, path, 0);
41
42     return 0;
43 }
```

Question 3

Recursive Path Finding

Given an $n \times n$ 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 (L) 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

```
1 #include <stdio.h>
2
3 int findPath(int matrix[100][100], int n, int x, int y, char* path,
4             int pathIndex) {
5     // If the destination is reached, print the path and return
6     if (x == n - 1 && y == n - 1) {
7         path[pathIndex] = '\0'; // Null terminate the string
8         printf("%s\n", path);
9         return 1;
10    }
11
12    int last = 'I';
13
14    if (pathIndex != 0)
15        last = path[pathIndex - 1];
16
17
18    // Try moving Right
19    if (last != 'L' && y + 1 < n && matrix[x][y + 1] == 1) {
20        path[pathIndex] = 'R';
21        if (findPath(matrix, n, x, y + 1, path, pathIndex + 1))
22            return 1;
23    }
24
25    // Try moving Left
26    if (last != 'R' && y - 1 >= 0 && matrix[x][y - 1] == 1) {
27        path[pathIndex] = 'L';
28        if (findPath(matrix, n, x, y - 1, path, pathIndex + 1))
29            return 1;
30    }
31
32    // Try moving Down
33    if (last != 'U' && x + 1 < n && matrix[x + 1][y] == 1) {
```

```

34     path[pathIndex] = 'D';
35     if(findPath(matrix, n, x + 1, y, path, pathIndex + 1))
36         return 1;
37 }
38
39 // Try moving Up
40 if (last != 'D' && x - 1 >= 0 && matrix[x - 1][y] == 1) {
41     path[pathIndex] = 'U';
42     if(findPath(matrix, n, x - 1, y, path, pathIndex + 1))
43         return 1;
44 }
45
46 return 0;
47 }
48
49 int main() {
50     int n;
51     scanf("%d", &n);
52
53     int matrix[100][100];
54     char path[1000];
55
56     // Read the matrix
57     for (int i = 0; i < n; i++) {
58         for (int j = 0; j < n; j++) {
59             scanf("%d", &matrix[i][j]);
60         }
61     }
62
63     findPath(matrix, n, 0, 0, path, 0);
64
65     return 0;
66 }

```