Assignment 6

Introduction to programming in C

Question 1

Given an n×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

```
#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;

return b;
</pre>
```

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

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>
   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
 5
       if (x = n - 1 & y = n - 1) {
   path[pathIndex] = '\0'; // Null terminate the string
6
            printf("%s\n", path);
            return;
9
10
       // Move Right
12
        if (y + 1 < n \&\& matrix[x][y + 1] == 1) {
13
            path[pathIndex] = 'R';
14
            findPath(matrix, n, x, y + 1, path, pathIndex + 1);
15
16
17
        // Move Down
18
        if (x + 1 < n \&\& matrix[x + 1][y] == 1)  {
19
            path[pathIndex] = \dot{D};
20
            findPath(matrix, n, x + 1, y, path, pathIndex + 1);
21
22
23 }
24
  int main() {
25
26
       int n;
        scanf("%d", &n);
27
28
        int matrix [100] [100];
29
       char\ path[200];\ //\ Assuming the path will not be longer than 2n
30
       -1 steps
31
        // Read the matrix
32
33
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
34
                 scanf("%d", &matrix[i][j]);
35
36
37
       \label{eq:continuous_path} find Path (\, matrix \,, \ n \,, \ 0 \,, \ 0 \,, \ path \,, \ 0) \,;
39
40
       return 0;
41
42 }
```

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 (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

```
#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) {
6
           path[pathIndex] = '\0'; // Null terminate the string
           printf("%s \n", path);
9
           return 1;
10
12
      int last = 'I';
13
       if (pathIndex !=0)
14
        last = path[pathIndex - 1];
15
16
17
       // Try moving Right
18
       if (last != 'L' && y + 1 < n && matrix[x][y + 1] == 1) {
19
           path[pathIndex] = 'R';
20
           if (findPath(matrix, n, x, y + 1, path, pathIndex + 1))
21
               return 1;
22
23
24
       // Try moving Left
25
       if (last != 'R' && y - 1 >= 0 && matrix[x][y - 1] == 1) {
26
           path[pathIndex] = 'L';
27
           if(findPath(matrix, n, x, y - 1, path, pathIndex + 1))
28
29
               return 1;
30
31
       // Try moving Down
32
       if (last != 'U' && x + 1 < n && matrix[x + 1][y] == 1) {
```

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