11) SHORTEST SAFE ROUTE PATH:

METHOD:

1) This sum is the same as that of the rat in the maze problem.Here, only the working condition is different.
2) Here we will start the recursion for a particular row if and only if the zeroth index position of that row is 1 and the adjacent values are also 1

   So function is like:

```
int minLength(vector<vector<int>>&matrix,int rows,int columns){
   int length=INT_MAX;
   for(int i=0;i<rows;i++){
      if(isSafe(matrix,i,0,rows,columns)){
         int temp=fun(matrix,i,0,0,rows,columns);
         length=min(length,temp);
      }
   }
   return length;
}
```

3) Now suppose we start for the particular row, then for moving ahead we have to check whether the adjacent of position at which we are moving does not contain zero.
4) So before moving at a particular position we check whether it is safe or not:

   The function is like:

```
bool isSafe(vector<vector<int>> &matrix,int i,int j,int rows,int columns){
   if(matrix[i][j]==0){
      return false;
   }
   int a[4]={-1,0,1,0};
   int b[4]={0,1,0,-1};
   for(int k=0;k<4;k++){
      if(i+a[k]<rows && i+a[k]>=0 && j+b[k]<columns && j+b[k]>=0 &&
matrix[i+a[k]][j+b[k]]==0){
         return false;
      }
   }
   return true;
}
```

5) Also if we got the safe position to move then we move to that position making the value of this position as 2,which means that we cannot again visit this position in backward direction.

   The main function is like:

```
int fun(vector<vector<int>> &matrix,int i,int j,int steps,int rows,int columns){
   if(matrix[i][j]==0){
      return INT_MAX;
   }
   else if(j==columns-1){
      return steps;
   }
   else{
      int a[4]={-1,0,1,0};
      int b[4]={0,1,0,-1};
```

```
            int length=INT_MAX;
            for(int k=0;k<4;k++){
                if(i+a[k]<rows && i+a[k]>=0 && j+b[k]<columns && j+b[k]>=0 &&
    matrix[i+a[k]][j+b[k]]!=2 && isSafe(matrix,i+a[k],j+b[k],rows,columns)==true){
                    matrix[i][j]=2;
                    int temp=fun(matrix,i+a[k],j+b[k],steps+1,rows,columns);
                    matrix[i][j]=1;
                    length=min(length,temp);
                }
            }
            return length;
        }
    }
```

THE COMPLETE CODE FOR THE ABOVE PROBLEM:

```
#include<iostream>
#include<vector>
#include<limits.h>
using namespace std;

bool isSafe(vector<vector<int>> &matrix,int i,int j,int rows,int columns){
    if(matrix[i][j]==0){
        return false;
    }
    int a[4]={-1,0,1,0};
    int b[4]={0,1,0,-1};
    for(int k=0;k<4;k++){
        if(i+a[k]<rows && i+a[k]>=0 && j+b[k]<columns && j+b[k]>=0 &&
matrix[i+a[k]][j+b[k]]==0){
            return false;
        }
    }
    return true;
}

int fun(vector<vector<int>> &matrix,int i,int j,int steps,int rows,int columns){
    if(matrix[i][j]==0){
        return INT_MAX;
    }
    else if(j==columns-1){
        return steps;
    }
    else{
        int a[4]={-1,0,1,0};
        int b[4]={0,1,0,-1};
        int length=INT_MAX;
        for(int k=0;k<4;k++){
            if(i+a[k]<rows && i+a[k]>=0 && j+b[k]<columns && j+b[k]>=0 &&
matrix[i+a[k]][j+b[k]]!=2 && isSafe(matrix,i+a[k],j+b[k],rows,columns)==true){
                matrix[i][j]=2;
                int temp=fun(matrix,i+a[k],j+b[k],steps+1,rows,columns);
                matrix[i][j]=1;
                length=min(length,temp);
            }
        }
        return length;
    }
}

int minLength(vector<vector<int>>&matrix,int rows,int columns){
    int length=INT_MAX;
    for(int i=0;i<rows;i++){
        if(isSafe(matrix,i,0,rows,columns)){
            int temp=fun(matrix,i,0,0,rows,columns);
            length=min(length,temp);
        }
    }
    return length;
}

int main(){
    vector<vector<int>> matrix={
        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 0, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 0, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 0, 1, 1, 1, 1 },
        { 1, 0, 1, 1, 1, 1, 1, 1, 0, 1 },
        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 0, 1, 1, 1, 1, 0, 1, 1, 1, 1 },
        { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 0, 1, 1, 1, 1, 1, 1 }
    };
    int length=minLength(matrix,12,10);
    cout<<"\n The length of the shortest path : "<<length;
    return 0;
}
```