Name: Implementation of Uniform Cost Search (UCS) to solve Weighted Route Finding Problem.

Description: In this problem you will be given an undirected weighted graph G, a source vertex (Initial State) S and a destination vertex D(Goal State). You need to calculate the shortest distance between S to D. In this problem, the shortest distance between two vertices means using the minimum weighted path or collection of edges to reach from one vertex to the other.

In the first line you will be given two integers V and E denoting the number of vertices and edges of this graph respectively. All the vertices will have ids between 1 to V. In the following E lines, you will get the information about all the edges. In the i^{th} line you will have three integers x (1<=x<=V), y (1<=y<=V) and w (1<=w<=100) which denotes there is an undirected edge between x and y. After E lines of input, you will get two integers S and D (1<=S<=V, 1<=D<=V) denoting the source and destination vertices of the problem.

As output, in the first line you will print an integer number d which will denote the minimum distance between S and D. If there lies no shortest path between S and D print -1.
In the following line print an integer P denoting the number of vertices that comprises a valid path satisfying the shortest path constraint (including source and destination). If there exists no valid shortest path print -1 instead (P = -1). In the case of a valid solution, print P lines afterward denoting the vertices that comprise the solution path.
After that print an integer denoting the number of explored nodes in the quest of reaching the destination node or goal state from source node or initial state.

Limits
1<=V<=100000, 1<=|E|<=100000
Test Cases:

Input	Output
4 5 1 2 1 2 4 10 1 4 11 1 3 1 3 4 2 1 4	3 3 1 3 4 3
8 10 1 7 1 1 8 1 2 3 2 2 4 5 2 5 2 2 6 1 3 4 2 4 5 3 4 8 1 6 7 1 2 4	4 3 2 3 4 6
15 25 1 5 3 1 9 7 1 13 11	9 3 2 9

2 13 5 2 10 1 2 9 8 2 14 18 3 8 8 3 7 1 3 4 3 3 13 12 4 12 1 4 13 19 4 9 3 5 6 18 6 14 11 6 10 2 7 12 10 8 9 1 8 12 19 9 13 14 10 11 18 10 13 18 11 15 16 14 15 17 2 8	8 5
large_in1.txt	532 18 249 475 252 158 199 265 427 128 502 723 569 808 780 557 922 720 849 742 883
large_in2.txt	311 8 7 385 215 162 1134 596 565 7232 1186

Name: Implementation of Uniform Cost Search (UCS) to solve Hill Climbing Problem.

Description:
An avenger is trying to climb a very dangerous hill. There are some places on the hill which are very steep and dangerous to climb. Also, there are some places on the hill which are pretty flat and possess no difficulties.

This hill climbing problem can be modeled as a 2D grid problem. Where the grid has M rows and N columns. Analogically here, M denotes the height and N denotes the width of the hill respectively.

The climber is at the bottom now. He will start climbing the hill aka start traversing the grid from the bottom row. In the grid, for each cell a value is given which denotes the danger lying in that cell. If the climber comes in this cell, it will add danger for him in the path to climb the hill. In this problem, you need to find the optimal path for the climber to reach the top of the hill, **minimizing** the total danger of climbing. From each cell, a climber can make three possible moves which has been shown in the following figure,

**climbed complete hill*

Can go here	Can go here	Can go here	
	Current cell		

**hill bottom*

In the first line, you will be given two values M and N denoting the number of rows and columns of the grid. Then you will have M lines of values where each line will have N values. Here the values lying in the i^{th} line denote the values of the i^{th} row of the grid. In such i^{th} line, j^{th} value denotes the danger value of grid cell (i,j).

In the output, first you need to print the minimum possible danger value for the climber to reach the top row. Outside of the bottom row means, he did not start climbing and outside of the top row means, he has finished climbing. The climber can not move right from any cells belonging to the rightmost column, similarly can not move left from the leftmost column, because it would bring death to him.
In the follwing line, you need to print the number of cells P that have been used to cross the hill (including source and destination) and the following P lines will contain the cells' information (row value and column value).
After that you need to print an integer number denoting the number of explored cells that this algorithm has calculated or expanded to reach the goal state. Remember that, a goal state is never expanded.

Limits
1<=M,N<=500
0<=Grid cell [i,j] <=100 for all (i,j) pairs

Test Cases:

Input	Output
3 3 2 10 0 5 0 7 0 6 4	0 3 2 0 1 1 0 2 2

4 3 0 2 1 9 8 2 3 2 7 10 5 2	7 4 3 2 2 1 1 2 0 2 4
3 3 14 25 0 1 5 10 5 1 2	6 3 2 1 1 1 0 2 5
3 4 1 4 4 4 4 4 1 4 2 1 4 3	6 3 2 1 1 1 0 0 7
10 10 12 96 71 28 53 50 24 83 99 30 73 70 37 97 99 1 32 99 84 13 73 13 15 49 69 50 96 3 55 17 32 77 47 41 51 75 3 80 44 87 16 20 2 64 20 53 97 54 17 93 14 7 99 72 66 4 4 84 4 28 47 57 56 12 86 65 10 19 59 99 51 61 31 96 7 5 31 37 86 6 78 37 55 86 99 11 72 13 17 51 87 86 32 11 52 1 78 1 34 76	146 10 9 5 8 5 7 5 6 6 5 5 4 5 3 6 2 7 1 6 0 6 62
large_in1.txt	Output not shown
large_out1.txt	Output not shown