South African Computer Olympiad Training DAY 2

Overview

Author	Carl	Bruce	IOI 1998
Problem	**Blow Torching walls**	**Gold**	**Polygon**
Program name	`walls.exe`	`gold.exe`	`polygon.exe`

Source name	`walls.pas`
	`walls.java`
	`walls.cpp`
		`gold.pas`
		`gold.java`
		`gold.cpp`
			`polygon.pas`
			`polygon.java`
			`polygon.cpp`

Input file	`stdin`	`stdin`	`stdin`
Output files	`stdout`	`stdout`	`stdout`
Time limit	5 seconds	5 seconds	1 second
Num. of tests	10	10	5
Points per test	10	10	20
Total points	**100**	**100**	**100**

The maximum total score for Day2 is 300 points.

Blow-torching Walls

Description:

After Farmer John's cows tried to escape the farm recently, Farmer John decided that he has had enough: these cows must be sent to the abbatoir! The cows, who have been spying on Farmer John, have discovered this... and are very worried. They _really_ need to escape now, otherwise they'll end up as roast beef on Farmer John's dinner table!
There is only one chance of escaping: and that is through the underground maze that runs beneath the farm. The cows raided Farmer John's office during the night and found a map of the maze. The only problem is that Farmer John has blocked up many parts of the maze with concrete wall... But luckily for the cows, they have also raided his toolshed and have stolen a blow-torch!
But blow-torching an entire piece of wall takes a _long_ time: so the cows need to work out the smallest number of walls that they'll have to blow-torch away in order to escape the maze, and also find the shortest route that needs this number of walls blow-torched away.

Input:

Your program should read its input from stdin. The first line of input will contain two integers, W and H, which indicate the width and height of the maze respectively. The second line of input will contain two integers, SX and SY, indicating the X and Y co-ordinates respectively of where the cows enter the maze. The third line of input will contain two integers, EX and EY, indicating the X and Y co-ordinates respectively of where the cows must exit the maze. The next H lines will each contain W integers, separated by spaces, that describe the maze. Each of these integers will either be 0 or 1: 0 indicates that there is no wall at that point in the maze, 1 indicates that there _is_ a wall.
The upper-left co-ordinate of the maze is (1,1) and the lower-right co-ordinate is (W,H). This is illustrated below:

```
(1,1)..........
...............
..........(W,H)
```

Task:

The cows can move from any co-ordinate in the maze to any directly adjacent co-ordinate (up, down, left or right). This counts as a distance of 1. If the cows move to a co-ordinate that has a wall, they blow-torch it first and then move to it.
1) Determine the smallest number of walls that need to be blow-torched for the cows to travel from (SX,SY) to (EX,EY).
2) Determine the shortest route from (SX,SY) to (EX,EY) that requires Z walls to be blow-torched, where Z is the number determined in part 1. There may be more than one such route: you only need to determine one of them.

Output:

Your program should write its output to stdout. The first line of output must contain 2 integers, Z and L. Z is the smallest number of walls that need to be blow-torched (determined in part 1 above), and L is the length of the shortest path that requires Z walls to be blow-torched. The next L+1 lines should each contain two integers, X and Y, which indicate the co-ordinates of the route that the cows must take.

Example:

INPUT
```
5 5
1 1
5 5
0 1 0 0 0
0 0 0 1 1
0 1 0 1 0
0 1 1 1 0
0 1 1 0 0
```

OUTPUT
```
1 8
1 1
1 2
2 2
3 2
3 3
4 3
5 3
5 4
5 5
```
So in this example, the cows must blow-torch 1 wall, and the shortest route that needs only 1 wall blow-torched has length 8.

Constraints:

1 <= W, H <= 1000
Co-ordinate (SX,SY) will never have a wall on it.
Time limit: 5 seconds

Scoring:

Let Z be the number of walls that your solution required to be blow-torched, and let Q be the optimal number. Let L be the length of th route that your solution found, and let M be the optimal number for a route requiring Q walls to be blow-torched. Put K = max(M, L). Your score is then given by:
S = (10 - [(Z - Q) * 3]) * [(M / K)^(Z - Q + 1)]
Thus if you find the optimal number of walls, and the optimal route for that number of walls, your solution scores 10 points :-)

Gold

You have won the lottery! However this lottery has an unusual twist: if you win, the six winning numbers are used to specify the dimensions of a solid rectangular volume of gold, which is your prize.

Task

The six numbers are grouped into three pairs; each of width describes one side of the box. The absolute difference of the two numbers is the length of that side. So for example if the numbers 1, 5, 6, 4, 8 and 2 are paired as (1, 6), (8, 2), (4, 5) then the box will have dimensions (6-1) x (8-2) x (5-4) = 5 x 6 x 1, giving it a volume of 30. However if they are paired as (1, 5), (6, 8), (4, 2) then the total volume will be 4 x 2 x 2 for an area of only 16.

Since the pairing of the numbers makes a difference to the final volume, you wish to choose the pairing that gives the largest possible volume. You must write a program that will compute the largest possible volume. To make it completely general, it should work in any number of dimensions, not just the usual three. So for example if the lottery has 10 numbers, then they will be put into 5 pairs and the absolute values of the differences in each pair are multiplied together.

Input data

The first line of input data is the number of lottery numbers, N. N will always be even. The next N lines contain the N lottery numbers as integers.

Sample input:

```
8
1
5
6
3
20
9
5
11
```

Output data:

The output is a single integer representing the maximum possible volume. Note that this value might not fit into a 32-bit integer.

Sample output:

```
2700
```

Constraints

```
2 <= N <= 200
0 <= each lottery number <= 1000
```

Time limit

5 seconds.

Polygon

Polygon is a game for one player that starts on a polygon with N vertices, like the one in Figure 1, where N=4. Each vertex is labeled with an integer and each edge is labeled with either the symbol + (addition) or the symbol * (product). The edges are numbered from 1 to N.

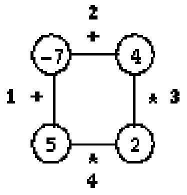

Figure 1. Graphical representation of a polygon

On the first move, one of the edges is removed.
Subsequent moves involve the following steps:
pick an edge E and the two vertices V_1 and V_2 that are linked by E; and
replace them by a new vertex, labeled with the result of performing the operation indicated in E on the labels of V_1 and V_2.
The game ends when there are no more edges, and its score is the label of the single vertex remaining.

Sample Game

Consider the polygon of Figure 1. The player started by removing edge 3. The effects are depicted in Figure 2.

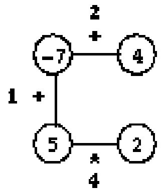

Figure 2. Removing edge 3
After that, the player picked edge 1,

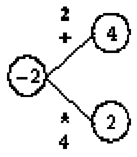

Figure 3. Picking edge 1
then edge 4,

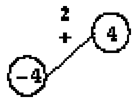

Figure 4. Picking edge 4
and, finally, edge 2. The score is 0.

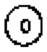

Figure 5. Picking edge 2

Task

Write a program that, given a polygon, computes the highest possible score and lists all the edges that, if removed on the first move, can lead to a game with that score.

Input Data

Input must be read from standard input, i.e. as you would read from keyboard. The input consists of two lines. On the first line is N, the number of vertices. The second line contains the labels of edges 1..N, interleaved with the vertices' labels (first that of the vertex between edges 1 and 2, then that of the vertex between edges 2 and 3, and so on, until that of the vertex between edges N and 1), all separated by one space. An edge label is either the letter `t` (representing +) or the letter `x` (representing *).

Sample Input

```
4
t -7 t 4 x 2 x 5
```

This is the input for Figure 1. The second line starts with the label of edge 1.

Output Data

Output must be written to standard out, i.e. as you would write to screen. On the first line of output your program must write the highest score one can get for the input polygon. On the second line it must write the list of all edges that, if removed on the first move, can lead to a game with that score. Edges must be written in increasing order, separated by one space.

Sample Output

```
33
1 2
```

This is the output for Figure 1.

Constraints

- 3 <= N <= 50
- For any sequence of moves, vertex labels are in the range [-32768,32767].

Time limit

1 second.