UW-Madison ACM ICPC Individual Contest

September 27, 2015

Setup

Before the contest begins, log in to your workstation and set up and launch the PC2 contest software using the following instructions. You will use this program to submit problem solutions, receive the judges' answers, and communicate clarification requests.

1. Download the custom PC2 package into a directory of your choosing from `www.cs.wisc.edu/~dieter/ICPC/15-16/pc2.tar.gz`
2. In a terminal window, cd to the directory where you downloaded the package and type `tar -xzvf pc2.tar.gz`
3. Type `cd pc2` followed by the command `bin/pc2team` – this brings up your PC2 terminal that will be your interface to the judges during the contest.
4. Log in using the login ID and password given to you by the judges when you arrive. They will be of the form teamX where X is an integer, and the password will be your UVa online judge username. If you have not yet sent your username to the judges, you will not be given a login and password by default – please see the judges to get one.

The Contest

Begin the contest by solving the problem on the next page **"count"**. This is a warmup problem designed to get you used to submitting problems via PC2. Code your solution to the problem and submit it as follows:

1. Click on the 'submit run' tab in your PC2 window.
2. In the dropdown menu labeled "Problem", choose **"count"**. Choose the programming language you used from the "language" dropdown menu. Then select your source code file by clicking the "select" button in the 'main file' section.

3. Submit your code by clicking "Submit" (note: clicking "Test" doesn't really do anything unless you've created your own test files, so don't expect it to automatically test your program against the sample input).
4. Wait – you will receive a judgment from the judge shortly by way of a pop-up window. If your answer comes back something other than "accepted", try again

The remaining problems are known to PC2 as **"pigs"**, **"server"**, **"wedding"**, and **"sums"**, respective of their order in this packet. All input comes from standard in, all output should be sent to standard out. You may use any online Java or C++ documentation, but not any other resource. You may use the printer at any time. Collaboration with others or searching the web for solutions to these problems is prohibited. Please turn off your cell phones. You may submit problem clarifications via the PC2 program at any time, but please read the problems thoroughly before doing so.

After the contest, please fill out the questionnaire on the next page, then join us in room 1325 for pizza and soda.

Information Form

Name: ____________________

CS Login: __________________________

Student status (e.g. Junior, first year grad student):
____

Year of birth: ________ Year starting college: ________

Which of C/C++/Java do you prefer? ____ Please indicate your proficiency in each language:

What classes have you taken (or are you taking) which are relevant to the ICPC?

What do you feel are your strengths with respect to the ICPC?

Are their other students you would like to be on a team with? Any you would not like to be on a team with?

Are you able to attend the World Finals in Phuket, Thailand, May 15-21, 2016 (and obtain a Visa and/or passport as necessary)?

Will you be in residence at UW-Madison in the Spring 2016 semester?

How many ICPC regionals have you participated in? ____ How many ICPC world finals? ____

If your team progresses to the world finals, how many hours per week could you commit to practicing? ________

Is there anything else we should know about you?

Warmup Problem: Count

Can you count from one up to any number N? Write a program to prove it!

Input

The input begins with a single number that describes the number of test cases. Each test case follows on its own line, and consists of a single positive integer $N \leq 1,000,000$ that describes how high you should count for that test case.

Output

The output for each test case should be on its own line, and consist of the numbers 1 through N (inclusive), each separated by a space.

Sample Input

```
3
3
5
10
```

Sample Output

```
1 2 3
1 2 3
1 2 3 4 5
1 2 3 4 5 6 7 8 9 10
```

The following pages contain four lettered problems. Please let me know if you do not see all four.

A: Pigs to Market

Farmer John has a pig farm near town A. He wants to visit his friend living in town B. During this journey he will visit n small villages so he decided to earn some money. He takes n pigs and plans to sell one pig in each village he visits.

Pork prices in villages are different, in the j-th village the people would buy pork at p_j rubles per kilogram. The distance from town A to the j-th village along the road to town B is d_j kilometers.

Pigs have different weights. Transporting one kilogram of pork per one kilometer of the road needs t rubles for additional fuel.

Help John decide which pig to sell in each town in order to earn as much money as possible.

Input

The first line of the input file contains integer numbers n ($1 \le n \le 1000$) and t ($1 \le t \le 10^9$). The second line contains n integer numbers w_i ($1 \le w_i \le 10^9$) – the weights of the pigs. The third line contains n integer numbers d_j ($1 \le d_j \le 10^9$) – the distances to the villages from the town A. The fourth line contains n integer numbers p_j ($1 \le p_j \le 10^9$) – the prices of pork in the villages.

Output

Output n numbers, the j-th number is the number pig to sell in the j-th village. The pigs are numbered from 1 in the order they are listed in the input file.

Sample Input

```
3 1
10 20 15
10 20 30
50 70 60
```

Sample Output

```
3 2 1
```

B: Server Transport

Michael has a powerful computer server that has hundreds of parallel processors and terabytes of main memory and disk space. Many important computations run continuously on this server, and power must be supplied to the server without interruption.

Michael's server must be moved to accommodate new servers that have been purchased recently. Fortunately, Michael's server has two redundant power supplies—as long as at least one of the two power supplies is connected to an electrical outlet, the server can continue to run. When the server is connected to an electrical outlet, it can be moved to any location which is not further away from the outlet than the length of the cord used to connect to the outlet.

Given which outlet Michael's server is plugged into initially and finally, and the locations of outlets in the server room, you should determine the smallest number of times you need to plug a cord into an electrical outlet in order to move the server while keeping the server running at all times. Note that, in the initial and final configuration, only one cord is connected to the power outlet.

Input

The first line of input is an integer giving the number of cases to follow. For each case, the first line is of the form:

$$n \ start \ end \ l1 \ l2$$

where:

- n is the number of outlets in the server room ($2 \leq n \leq 1000$).
- *start* is the index (starting from 1) of the outlet the server is initially connected to.
- *end* is the index (starting from 1) of the outlet the server is finally connected to.
- $l1$ and $l2$ are the positive lengths of the two power cords, with at most three digits of precision after the decimal point ($0 < l1, l2 \leq 30000$).

These are followed by n lines giving the integer coordinates of the wall outlets, one per line, with the kth line giving the location of the kth outlet. All coordinates are specified as two integers (x and y coordinates) separated by a space, with absolute values at most 30000. You may assume that all coordinates are distinct, and that the initial outlet and the final outlet are different.

Output

For each case, print the minimum number of times you need to plug a cord into an electrical outlet in order to move the server to the final location while keeping the server running at all times. If this is not possible, print "Impossible".

Sample Input

```
2
4 1 4 2.000 1.000
0 0
0 4
4 0
4 4
9 1 4 2.000 3.000
0 7
-6 2
-3 3
6 2
-6 -3
3 -3
6 -3
-3 -7
0 -7
```

Sample Output

```
Impossible
8
```

C: Wedding Arrangements

Bob and Alice are getting married, and the party planning has begun! Every competitive programming enthusiast has been invited to the celebration, and everyone will be sitting in a single row at an extremely long dining table.

However, Bob and Alice still can't agree on a seating chart for all of their very many guests. The lack of cooperation has the wedding party worried. Help decide if Bob and Alice are right for each other by determining the number of distinct pairs of guests who appear in different orders in the two charts.

Input

The input file will contain multiple test cases. Each test case begins with a single line containing an integer n ($1 \leq n \leq 100,000$) indicating the number of wedding guests. The next two lines represent Bob and Alice's seating charts, respectively. Each seating chart is specified as a single line of n unique alphabetical strings; the set of strings in each line are guaranteed to be identical. The end-of-input is denoted by a line containing the number 0.

Output

For each input test case, output a single integer denoting, out of the $\binom{n}{2}$ distinct pairs of wedding guests, how many pairs appear in different orders in Bob and Alice's seating arrangements.

Sample Input

```
3
Tourist Bob Alice
Bob Alice Tourist
5
A B C D E
B A D E C
0
```

Sample Output

```
2
3
```

D: Sums

Given an integer N, express it as the sum of at least two consecutive positive integers. For example:

- $10 = 1 + 2 + 3 + 4$
- $24 = 7 + 8 + 9$

If there are multiple solutions, output the one with the smallest possible number of summands.

Input

The first line of input contains the number of test cases T. Each test case consists of one line containing an integer N ($1 \leq N \leq 10^9$).

Output

For each test case, output a single line containing the equation in the format:

$$N = a + (a + 1) + \ldots + b$$

as in the example. If there is no solution, output a single word IMPOSSIBLE instead.

Sample Input

```
3
8
10
24
```

Sample Output

```
IMPOSSIBLE
10 = 1 + 2 + 3 + 4
24 = 7 + 8 + 9
```