Problem Set for JAG Practice Contest 2007

Japanese Alumni Group

Contest Held on 21 Oct 2007

Status of Problems

All problems were newly created by the members of Japanese Alumni Group.

The sample input in Problem C was found to contain an illegal dataset just before the contest began. It is removed in this PDF file.

This problem set was typeset by LaTeX 2_ε with a style file made by a member of JAG from scratch, so that the statement looks like those used in ACM-ICPC Regional Contest held in Japan.

Terms of Use

Problem A
Space Coconut Crab II
Input: A.txt

A space hunter, Ken Marineblue traveled the universe, looking for the space coconut crab. The space coconut crab was a crustacean known to be the largest in the universe. It was said that the space coconut crab had a body of more than 400 meters long and a leg span of no shorter than 1000 meters long. Although there were numerous reports by people who saw the space coconut crab, nobody have yet succeeded in capturing it.

After years of his intensive research, Ken discovered an interesting habit of the space coconut crab. Surprisingly, the space coconut crab went back and forth between the space and the hyperspace by phase drive, which was the latest warp technology. As we, human beings, was not able to move to the hyperspace, he had to work out an elaborate plan to capture them. Fortunately, he found that the coconut crab took a long time to move between the hyperspace and the space because it had to keep still in order to charge a sufficient amount of energy for phase drive. He thought that he could capture them immediately after the warp-out, as they moved so slowly in the space.

He decided to predict from the amount of the charged energy the coordinates in the space where the space coconut crab would appear, as he could only observe the amount of the charged energy by measuring the time spent for charging in the hyperspace. His recent spaceship, Weapon Breaker, was installed with an artificial intelligence system, CANEL. She analyzed the accumulated data and found another surprising fact; the space coconut crab always warped out near to the center of a triangle that satisfied the following conditions:

- each vertex of the triangle was one of the planets in the universe;
- the length of every side of the triangle was a prime number; and
- the total length of the three sides of the triangle was equal to T, the time duration the space coconut crab had spent in charging energy in the hyperspace before moving to the space.

CANEL also devised the method to determine the three planets comprising the triangle from the amount of energy that the space coconut crab charged and the lengths of the triangle sides. However, the number of the candidate triangles might be more than one.

Ken decided to begin with calculating how many different triangles were possible, analyzing the data he had obtained in the past research. Your job is to calculate the number of different triangles which satisfies the conditions mentioned above, for each given T.

Input

The input consists of multiple datasets. Each dataset comes with a line that contains a single positive integer T ($1 \leq T \leq 30000$).

The end of input is indicated by a line that contains a zero. This should not be processed.

Output

For each dataset, print the number of different possible triangles in a line. Two triangles are different if and only if they are not congruent.

Sample Input

```
10
12
15
777
4999
5000
0
```

Output for the Sample Input

```
0
1
2
110
2780
0
```

Problem B
Sort the Panels
Input: B.txt

There was an explorer Henry Nelson traveling all over the world. One day he reached an ancient building. He decided to enter this building for his interest, but its entrance seemed to be locked by a strange security system.

There were some black and white panels placed on a line at equal intervals in front of the entrance, and a strange machine with a cryptogram attached. After a while, he managed to read this cryptogram: this entrance would be unlocked when the panels were rearranged in a certain order, and he needed to use this special machine to change the order of panels.

All he could do with this machine was to swap arbitrary pairs of panels. Each swap could be performed by the following steps:

- move the machine to one panel and mark it;
- move the machine to another panel and again mark it; then
- turn on a special switch of the machine to have the two marked panels swapped.

It was impossible to have more than two panels marked simultaneously. The marks would be erased every time the panels are swapped.

He had to change the order of panels by a number of swaps to enter the building. Unfortunately, however, the machine was so heavy that he didn't want to move it more than needed. Then which steps could be the best?

Your task is to write a program that finds the minimum cost needed to rearrange the panels, where moving the machine between adjacent panels is defined to require the cost of one. You can arbitrarily choose the initial position of the machine, and don't have to count the cost for moving the machine to that position.

Input

The input consists of multiple datasets.

Each dataset consists of three lines. The first line contains an integer N, which indicates the number of panels ($2 \leq N \leq 16$). The second and third lines contain N characters each, and describe the initial and final orders of the panels respectively. Each character in these

descriptions is either 'B' (for black) or 'W' (for white) and denotes the color of the panel. The panels of the same color should *not* be distinguished.

The input is terminated by a line with a single zero.

Output

For each dataset, output the minimum cost on a line. You can assume that there is at least one way to change the order of the panels from the initial one to the final one.

Sample Input

```
4
WBWB
BWBW
8
WWWWBWBB
WWBBWBWW
0
```

Output for the Sample Input

```
3
9
```

Problem C
Disarmament of the Units
Input: C.txt

This is a story in a country far away. In this country, two armed groups, ACM and ICPC, have fought in the civil war for many years. However, today October 21, reconciliation was approved between them; they have marked a turning point in their histories.

In this country, there are a number of roads either horizontal or vertical, and a town is built on every endpoint and every intersection of those roads (although only one town is built at each place). Some towns have units of either ACM or ICPC. These units have become unneeded by the reconciliation. So, we should make them disarm in a verifiable way simultaneously. All disarmament must be carried out at the towns designated for each group, and only one unit can be disarmed at each town. Therefore, we need to move the units.

The command of this mission is entrusted to you. You can have only one unit moved by a single order, where the unit must move to another town directly connected by a single road. You cannot make the next order to any unit until movement of the unit is completed. The unit cannot pass nor stay at any town another unit stays at. In addition, even though reconciliation was approved, members of those units are still irritable enough to start battles. For these reasons, two units belonging to different groups may not stay at towns on the same road. You further have to order to the two groups alternatively, because ordering only to one group may cause their dissatisfaction. In spite of this restriction, you can order to either group first.

Your job is to write a program to find the minimum number of orders required to complete the mission of disarmament.

Input

The input consists of multiple datasets. Each dataset has the following format:

$n\ m_A\ m_I$
$rx_{1,1}\ ry_{1,1}\ rx_{1,2}\ ry_{1,2}$
...
$rx_{n,1}\ ry_{n,1}\ rx_{n,2}\ ry_{n,2}$
$sx_{A,1}\ sy_{A,1}$
...
$sx_{A,m_A}\ sy_{A,m_A}$
$sx_{I,1}\ sy_{I,1}$
...

sx_{I,m_I} sy_{I,m_I}
$tx_{A,1}$ $ty_{A,1}$
...
tx_{A,m_A} ty_{A,m_A}
$tx_{I,1}$ $ty_{I,1}$
...
tx_{I,m_I} ty_{I,m_I}

n is the number of roads. m_A and m_I are the number of units in ACM and ICPC respectively ($m_A > 0$, $m_I > 0$, $m_A + m_B < 8$). $(rx_{i,1}, ry_{i,1})$ and $(rx_{i,2}, ry_{i,2})$ indicate the two endpoints of the i-th road, which is always parallel to either x-axis or y-axis. A series of $(sx_{A,i}, sy_{A,i})$ indicates the coordinates of the towns where the ACM units initially stay, and a series of $(sx_{I,i}, sy_{I,i})$ indicates where the ICPC units initially stay. A series of $(tx_{A,i}, ty_{A,i})$ indicates the coordinates of the towns where the ACM units can be disarmed, and a series of $(tx_{I,i}, ty_{I,i})$ indicates where the ICPC units can be disarmed.

You may assume the following:

- the number of towns, that is, the total number of coordinates where endpoints and/or intersections of the roads exist does not exceed 18;
- no two horizontal roads are connected nor ovarlapped; no two vertical roads, either;
- all the towns are connected by roads; and
- no pair of ACM and ICPC units initially stay in the towns on the same road.

The input is terminated by a line with triple zeros, which should not be processed.

Output

For each dataset, print the minimum required number of orders in a line. It is guaranteed that there is at least one way to complete the mission.

Sample Input

```
5 1 1
0 0 2 0
1 1 2 1
1 2 3 2
1 0 1 2
2 0 2 2
0 0
3 2
2 1
```

```
1 2
2 1 1
1 0 1 2
0 1 2 1
1 0
0 1
1 0
2 1
2 1 1
1 0 1 2
0 1 2 1
1 0
0 1
1 2
2 1
4 1 1
0 0 2 0
1 1 3 1
1 0 1 1
2 0 2 1
0 0
3 1
1 0
2 1
5 1 1
0 0 2 0
1 1 2 1
1 2 3 2
1 0 1 2
2 0 2 2
0 0
3 2
3 2
0 0
8 3 3
1 1 3 1
0 2 4 2
1 3 5 3
2 4 4 4
1 1 1 3
2 0 2 4
3 1 3 5
4 2 4 4
0 2
1 1
```

```
2 0
3 5
4 4
5 3
3 5
4 4
5 3
0 2
1 1
2 0
0 0 0
```

Output for the Sample Input

```
3
1
2
2
7
18
```

Problem D
So Sleepy
Input: D.txt

You have an appointment to meet a friend of yours today, but you are so sleepy because you didn't sleep well last night.

As you will go by trains to the station for the rendezvous, you can have a sleep on a train. You can start to sleep as soon as you get on a train and keep asleep just until you get off. However, because of your habitude, you can sleep only on one train on your way to the destination.

Given the time schedule of trains as well as your departure time and your appointed time, your task is to write a program which outputs the longest possible time duration of your sleep in a train, under the condition that you can reach the destination by the appointed time.

Input

The input consists of multiple datasets. Each dataset looks like below:

$S\ T$
$D\ Time_D\ A\ Time_A$
N_1
$K_{1,1}\ Time_{1,1}$
...
$K_{1,N_1}\ Time_{1,N_1}$
N_2
$K_{2,1}\ Time_{2,1}$
...
$K_{2,N_2}\ Time_{2,N_2}$
...
N_T
$K_{T,1}\ Time_{T,1}$
...
$K_{T,N_T}\ Time_{T,N_T}$

The first line of each dataset contains S ($1 \leq S \leq 1000$) and T ($0 \leq T \leq 100$), which denote the numbers of stations and trains respectively. The second line contains the departure station (D), the departure time ($Time_D$), the appointed station (A), and the appointed time ($Time_A$),

in this order. Then T sets of time schedule for trains follow. On the first line of the i-th set, there will be N_i which indicates the number of stations the i-th train stops at. Then N_i lines follow, each of which contains the station identifier ($K_{i,j}$) followed by the time when the train stops at (i.e. arrives at and/or departs from) that station ($Time_{i,j}$).

Each station has a unique identifier, which is an integer between 1 and S. Each time is given in the format $hh:mm$, where hh represents the two-digit hour ranging from "`00`" to "`23`", and mm represents the two-digit minute from "`00`" to "`59`".

The input is terminated by a line that contains two zeros.

You may assume the following:

- each train stops at two stations or more;
- each train never stops at the same station more than once;
- a train takes at least one minute from one station to the next;
- all the times that appear in each dataset are those of the same day; and
- as being an expert in transfer, you can catch other trains that depart at the time just you arrive the station.

Output

For each dataset, your program must output the maximum time in minutes you can sleep if you can reach to the destination station in time, or "`impossible`" (without the quotes) otherwise.

Sample Input

```
3 1
1 09:00 3 10:00
3
1 09:10
2 09:30
3 09:40
3 2
1 09:00 1 10:00
3
1 09:10
2 09:30
3 09:40
3
3 09:20
2 09:30
```

```
1 10:00
1 0
1 09:00 1 10:00
1 0
1 10:00 1 09:00
3 1
1 09:00 3 09:35
3
1 09:10
2 09:30
3 09:40
4 3
1 09:00 4 11:00
3
1 09:10
2 09:20
4 09:40
3
1 10:30
3 10:40
4 10:50
4
1 08:50
2 09:30
3 10:30
4 11:10
0 0
```

Output for the Sample Input

```
30
30
0
impossible
impossible
60
```

Problem E
Alice in Foxland
Input: E.txt

> — *The world is made of foxes; people with watchful eyes will find foxes in every fragment of the world.*

Defoxyrenardnucleic Acids (or DNAs for short) and *Renardnucleic Acids* (or RNAs) are similar organic molecules contained in organisms. Each of those molecules consists of a sequence of 26 kinds of nucleobases. We can represent such sequences as strings by having each lowercase letter from '`a`' to '`z`' denote one kind of nucleobase. The only difference between DNAs and RNAs is the way of those nucleobases being bonded.

In the year of 2323, Prof. Fuchs discovered that DNAs including particular substrings act as catalysts, that is, accelerate some chemical reactions. He further discovered that the reactions are accelerated as the DNAs have longer sequences. For example, DNAs including the sequence "`fox`" act as catalysts that accelerate dissolution of nitrogen oxides (NO_x). The DNAs "`redfox`" and "`cutefoxes`" are two of the instances, and the latter provides more acceleration than the former. On the other hand, the DNA "`foooox`" will *not* be such a catalyst, since it does not include "`fox`" as a *substring*.

DNAs can be easily obtained by extraction from some plants such as glory lilies. However, almost all of extracted molecules have different sequences each other, and we can obtain very few molecules that act as catalysts. From this background, many scientists have worked for finding a way to obtain the demanded molecules.

In the year of 2369, Prof. Hu finally discovered the following process:

1. Prepare two DNAs X and Y.
2. Generate RNAs X' and Y' with their sequences copied from the DNAs X and Y respectively.
3. Delete zero or more bases from X' using some chemicals to obtain an RNA X''.
4. Also delete zero or more bases from Y' to obtain an RNA Y'', which has the same sequence as X''.
5. Obtain a resulting DNA Z from the two RNAs X'' and Y'' in such a way that Z have the same sequence as X'' and Y''.

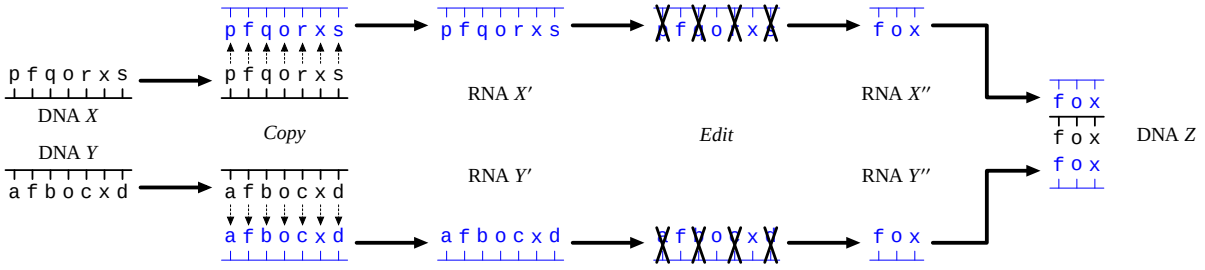


Figure 1: New Method for Generation of DNAs

The point is use of RNAs. It is difficult to delete specific bases on DNAs but relatively easy on RNAs. On the other hand, since RNAs are less stable than DNAs, there is no known way to obtain RNAs directly from some organisms. This is why we obtain RNAs from DNAs.

Alice is a researcher in Tail Environmental Natural Catalyst Organization. She is now requested to generate DNAs with particular substrings, applying Hu's method to some pairs of DNAs. Since longer DNAs are preferred, she wants a means of knowing the longest possible DNAs. So she asked you for help.

Your task is to write a program that outputs the sequence of the longest possible DNA which can be generated from the given pair of DNAs X and Y and contains the given substring C.

Input

The input consists of multiple datasets. Each dataset consists of three lines. The first and second lines contain the sequences X and Y respectively. The third line contains the substring C. Each of the sequences and the substrings contains only lowercase letters ('`a`'–'`z`') and has the length between 1 and 1600 inclusive.

The input is terminated by a line with "`*`".

Output

For each dataset, output the longest sequence in a line. If DNAs with the substring C cannot be obtained in any way, output "`Impossible`" instead. Any of them is acceptable in case there are two or more longest DNAs possible.

Sample Input

```
czujthebdgfoliskax
bdgchuptzefliskaox
fox
fennec
oinarisama
xiaorong
```

```
customizedrevisedfoooox
accurateredundantfooooooooox
fox
*
```

Output for the Sample Input

```
cutefox
Impossible
cuteredfox
```

Problem F
Lying about Your Age
Input: F.txt

You have moved to a new town. As starting a new life, you have made up your mind to do one thing: lying about your age. Since no person in this town knows your history, you don't have to worry about immediate exposure. Also, in order to ease your conscience somewhat, you have decided to claim ages that represent your real ages when interpreted as the base-M numbers ($2 \leq M \leq 16$). People will misunderstand your age as they interpret the claimed age as a decimal number (i.e. of the base 10).

Needless to say, you don't want to have your real age revealed to the people in the future. So you should claim only one age each year, it should contain digits of decimal numbers (i.e. '0' through '9'), and it should always be equal to or greater than that of the previous year (when interpreted as decimal).

How old can you claim you are, after some years?

Input

The input consists of multiple datasets. Each dataset is a single line that contains three integers A, B, and C, where A is the present real age (in decimal), B is the age you presently claim (which can be a non-decimal number), and C is the real age at which you are to find the age you will claim (in decimal). It is guaranteed that $0 \leq A < C \leq 200$, while B may contain up to eight digits. No digits other than '0' through '9' appear in those numbers.

The end of input is indicated by $A = B = C = -1$, which should not be processed.

Output

For each dataset, print in a line the minimum age you can claim when you become C years old. In case the age you presently claim cannot be interpreted as your real age with the base from 2 through 16, print -1 instead.

Sample Input

```
23 18 53
46 30 47
-1 -1 -1
```

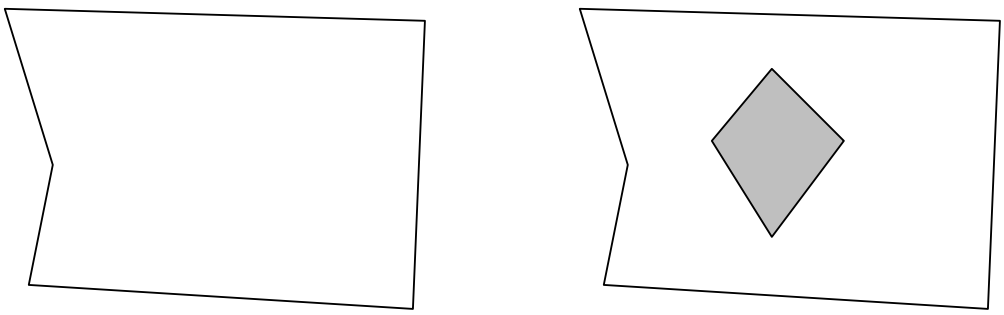
Output for the Sample Input

```
49
-1
```

Problem G
Turn Polygons
Input: G.txt

HCII, the health committee for interstellar intelligence, aims to take care of the health of every interstellar intelligence.

Staff of HCII uses a special equipment for health checks of patients. This equipment looks like a polygon-shaped room with plenty of instruments. The staff puts a patient into the equipment, then the equipment rotates clockwise to diagnose the patient from various angles. It fits many species without considering the variety of shapes, but not suitable for big patients. Furthermore, even if it fits the patient, it can hit the patient during rotation.



(rough sketch) (with a patient)

Figure 2: The Equipment used by HCII

The interior shape of the equipment is a polygon with M vertices, and the shape of patients is a *convex* polygon with N vertices. Your job is to calculate how much you can rotate the equipment clockwise without touching the patient, and output the angle in degrees.

Input

The input consists of multiple data sets, followed by a line containing "`0 0`". Each data set is given as follows:

$M\ N$
$px_1\ py_1\ px_2\ py_2\ \ldots\ px_M\ py_M$
$qx_1\ qy_1\ qx_2\ qy_2\ \ldots\ qx_N\ qy_N$
$cx\ cy$

The first line contains two integers M and N ($3 \leq M, N \leq 10$). The second line contains $2M$ integers, where (px_j, py_j) are the coordinates of the j-th vertex of the equipment. The third line contains $2N$ integers, where (qx_j, qy_j) are the coordinates of the j-th vertex of the patient. The fourth line contains two integers cx and cy, which indicate the coordinates of the center of rotation.

All the coordinates are between -1000 and 1000, inclusive. The vertices of each polygon are given in counterclockwise order.

At the initial state, the patient is inside the equipment, and they don't intersect each other. You may assume that the equipment doesn't approach patients closer than 10^{-6} without hitting patients, and that the equipment gets the patient to stick out by the length greater than 10^{-6} whenever the equipment keeps its rotation after hitting the patient.

Output

For each data set, print the angle in degrees in a line. Each angle should be given as a decimal with an arbitrary number of fractional digits, and with an absolute error of at most 10^{-7}.

If the equipment never hit the patient during the rotation, print 360 as the angle.

Sample Input

```
5 4
0 0 20 0 20 10 0 10 1 5
10 3 12 5 10 7 8 5
10 5
4 3
0 0 10 0 10 5 0 5
3 1 6 1 5 4
0 0
5 3
0 0 10 0 10 10 0 10 3 5
1 1 4 1 4 6
0 0
0 0
```

Output for the Sample Input

```
360.0000000
12.6803835
3.3722867
```

Problem H
Robots' Crash
Input: H.txt

Prof. Jenifer A. Gibson is carrying out experiments with many robots. Since those robots are expensive, she wants to avoid their crashes during her experiments at her all effort. So she asked you, her assistant, as follows.

"Suppose that we have n ($2 \leq n \leq 100000$) robots of the circular shape with the radius of r, and that they are placed on the xy-plane without overlaps. Each robot starts to move straight with a velocity of either v or $-v$ simultaneously, say, at the time of zero. The robots keep their moving infinitely unless I stop them. I'd like to know in advance if some robots will crash each other. The robots crash when their centers get closer than the distance of $2r$. I'd also like to know the time of the first crash, if any, so I can stop the robots before the crash. Well, could you please write a program for this purpose?"

Input

The input consists of multiple datasets. Each dataset has the following format:

n
$vx\ vy$
r
$rx_1\ ry_1\ u_1$
$rx_2\ ry_2\ u_2$
$\ldots$
$rx_n\ ry_n\ u_n$

n is the number of robots. (vx, vy) denotes the vector of the velocity v. r is the radius of the robots. (rx_i, ry_i) denotes the coordinates of the center of the i-th robot. u_i denotes the moving direction of the i-th robot, where 1 indicates the i-th robot moves with the velocity (vx, vy), and -1 indicates $(-vx, -vy)$.

All the coordinates range from -1200 to 1200. Both vx and vy range from -1 to 1.

You may assume all the following hold for any pair of robots:

- they must not be placed closer than the distance of $(2r + 10^{-8})$ at the initial state;

- they must get closer than the distance of $(2r - 10^{-8})$ if they crash each other at some point of time; and
- they must not get closer than the distance of $(2r + 10^{-8})$ if they don't crash.

The input is terminated by a line that contains a zero. This should not be processed.

Output

For each dataset, print the first crash time in a line, or "`SAFE`" if no pair of robots crashes each other. Each crash time should be given as a decimal with an arbitrary number of fractional digits, and with an absolute error of at most 10^{-4}.

Sample Input

```
2
1.0 0.0
0.5
0.0 0.0 1
2.0 0.0 -1
2
1.0 0.0
0.5
0.0 0.0 -1
2.0 0.0 1
0
```

Output for the Sample Input

```
0.500000
SAFE
```

Problem I
Linear Ether Geometry
Input: I.txt

Euclid, dwelt in Alexandria, decided to buy and move into a new house as his study was becoming cramped because of too many books he had bought.

His new house had many libraries. His first task was to have all the libraries connected to the Internet. As shown in Figure 3, all the libraries were located along one side of a straight hallway, and each of them had a connector to the inside on the wall (indicated by ∘ in the figure). He had already finished wiring inside each library according to his great idea, so next he was to connect between the libraries and the Internet. The only connector to the Internet was located at the end of the hallway (indicated by • in the figure).

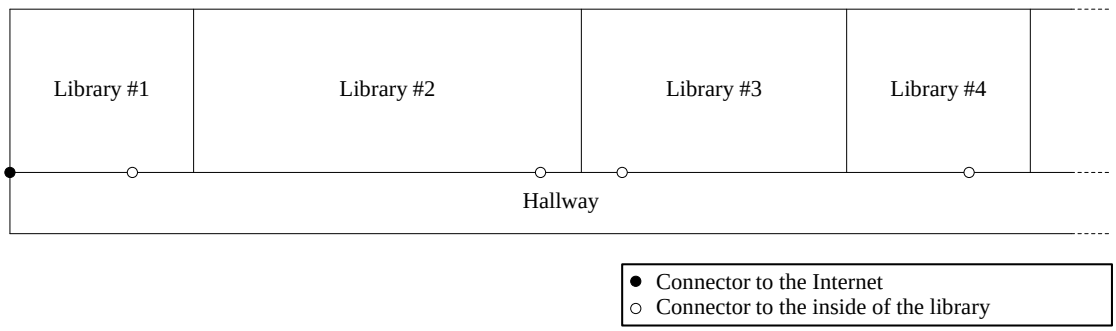


Figure 3: The hallway and the libraries

He had brought several Ethernet cables and enough number of hubs with sufficient ports from his old house. Naturally, he wanted to connect all libraries to the Internet with them. In addition, he wanted to lay them straight along the wall, as illustrated in Figure 4, with the following criteria: first to minimize the number of used hubs, and then to make the slack of the cable as less as possible (i.e. to minimize total extra lengths of the cables).

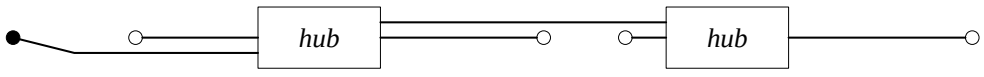


Figure 4: Laying cables along the wall

Your mission is to answer the minimum number of hubs and the minimum total extra lengths of the cables for given situations. You may assume that the sizes of hubs and the thicknesses of the cables are both negligible.

Input

The input consists of multiple datasets. Each dataset has the following format:

$N\ M\ L$
$x_1\ x_2\ \ldots\ x_N$
$l_1\ l_2\ \ldots\ l_M$

The first line contains three integers, where N ($1 \leq N \leq 5$) denotes the number of libraries, M ($1 \leq M \leq 10$) denotes the number of cables, and L ($1 \leq L \leq 20$) denotes the length of hallway. The second line contains N positive integers in their increasing order, where the i-th integer indicates the x-coordinate of the connector to the i-th library. The third line contains M positive integers in their increasing order, where the i-th integer indicates the length of the i-th cable. No cable has the length greater than L.

The input is terminated by a line that contains triple zeros. This is not part of any data sets.

Output

For each data set, print two integers on a line. The first integer should represent the minimum number of hubs, and the second should represent the minimum total extra lengths of the cables. These two integers should be separated by a single space.

If there is no possible layout, print "`Impossible`" instead.

Sample Input

```
2 4 10
5 10
1 1 3 9
3 5 10
4 6 10
2 2 2 4 5
3 4 10
3 6 10
2 3 4 5
3 5 10
2 3 4
1 2 3 4 5
4 9 20
5 10 15 20
5 5 5 6 6 6 7 7 7
5 10 20
4 9 13 17 20
1 1 1 1 1 1 1 20 20 20
0 0 0
```

Output for the Sample Input

```
2 0
2 1
Impossible
1 0
2 8
5 17
```