BENELUX ALGORITHM PROGRAMMING CONTEST

LEIDEN

Powered by Calco-IT

Preliminaries

2 October 2010

Problems

Problems Preliminaries BAPC 2010

A – Evolution
B – Have a Nice Day
C – Serial Numbers
D – Equal Is Not Really Equal
E – The Great Cleanup
F – Stock Market
G – Acrobat Reader
H – Farmer John
I – Imagine
J – My Cousin Obama

Output of Real Numbers

In problems A and H, the output consists of real numbers, which must be rounded and displayed to six digits after the decimal point. This output format is strict. Your output must be identical to that of the jury.

There are probably many ways to print a real number in the required format. If you do not know by heart how to do this, then the following examples may be useful.

In C

```
double x;

printf ("%.6lf\n", x);
```

In C++

```
#include <iomanip>

double x;

cout << setiosflags (ios::fixed) << setprecision (6) << x << '\n';
```

In Java

```
import java.text.*;

double x;

DecimalFormat fm = new DecimalFormat ("0.000000", new DecimalFormatSymbols(Locale.US));
System.out.println (fm.format(x));
```

A Evolution

Problem

Dr. Beverly has been experimenting with an interesting kind of organism. This organism's DNA consists of a single string of k letters over some alphabet of size d. One hour after its birth, an individual gives birth to one offspring, after which it happily lives on for another 15 minutes or so. This process repeats until there is no cold pizza left to feed on.

A *mutation* is a replacement of a character in the string with any character from the alphabet, possibly with itself. Mutations may occur from one generation to the next. The probability of a mutation is the same for each of the characters of the alphabet and is denoted by p.

Unfortunately, after Dr. Beverly started an experiment with one such creature, she got so caught up in a computer game of some kind that she forgot all about her experiment. When she came back a while later, she found the remains of N creatures. She sampled their DNA, hoping she would be able to figure out which of the corpses belongs to the creature she started the experiment with. Oh if only she had made notes! Anyway, given N strings of DNA, can you compute the probability for each individual that it was the original creature? We may assume that each of the N strings is (a priori) equally likely to serve as the initial string. The order in which the N strings are given is random.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with three integers N, k and d, satisfying $1 \leq N \leq 15$, $1 \leq k \leq 8$, $1 \leq d \leq 4$ and a real number p, satisfying $0.2 \leq p \leq 0.5$. The numbers are separated by single spaces.
- N lines, each with a string of length k over some alphabet of size d. This alphabet is a subset of { A, B, C, ..., Z } and is the same for all N strings. The string on the i-th line represents the DNA of the i-th creature.

Output

For every test case in the input, the output should contain N lines, each with a real number, rounded and displayed to six digits after the decimal point. The number on the i-th line should be the probability that the i-th creature in the input was the original creature.

Example

Input	Output
2	0.466667
3 1 2 0.25	0.266667
A	0.266667
C	0.046602
C	0.393710
4 4 4 0.50	0.083333
GTTG	0.476354
TGTG	
TTTG	
GTGT	

This page intentionally left blank.

B Have a Nice Day

Problem

Rumour has it that the $\mathcal{P}$ versus $\mathcal{NP}$ question has been solved: the two classes are not equal. This implies that many well-known problems, such as the Traveling Salesman Problem, will remain difficult forever. It can be considered a waste of time to search for polynomial time solutions: essentially only brute-force approaches can solve them. Nothing you can do about that.

In view of the international crisis, the new Dutch government has therefore announced that on certain days it is not allowed to work on these hard problems anymore. Instead, one must concentrate on easier issues. These days are called *nice*. Of course, the algorithm to decide whether a given date is nice or not should itself be easy. So far politicians could not find such an algorithm. Can you?

A date *day month year* is written down using the digits 0,...,9. A date is called *nice* if the digits occurring in it occur an equal number of times, *and* if it can be *split*. A date can be split if its *four number set* can be divided into two disjoint subsets with equal sum; the four numbers are the day, the month, the left part of the year (the number represented by its first and second digit; for 1957 this is 19) and the right part of the year (the number represented by its third and fourth digit; for 2000 this is 0). For example, 16 5 4928 is nice, because all digits occur exactly once and $16 + 5 + 28 = 49$.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with three integers D, M and Y separated by single spaces, satisfying $1 \leq D \leq 31$, $1 \leq M \leq 12$ and $1000 \leq Y \leq 9999$: the day, month and year of a valid date, respectively. There are no leading zeros; e.g., June is represented as 6 and not as 06.

Output

For every test case in the input, the output should contain the string `"yes"` or `"no"`: the fact whether the date is nice or not.

Example

Input	Output

`4`
`16 5 4928`
`14 12 2747`
`11 11 1111`
`3 3 2014`
	`yes`
	`no`
	`yes`
	`no`

This page intentionally left blank.

C Serial Numbers

Problem

The great guitar player Bernhard-Anton Peter Cobain is playing with his band the Awesome Crying Metalheads in the wonderful city of Leiden. The show goes well and Bernhard-Anton is very happy, but after the show he has a big problem: his guitars are mixed up with all the guitars of the other guitar players, and Bernhard-Anton does not remember which guitars were his.

Fortunately, every guitar has a unique serial number which is an integer between 1 and $100,000$. The only thing that Bernhard-Anton remembers is that the sum of all the serial numbers of his guitars is a multiple of M. Given all serial numbers and the number M, calculate the maximum number of guitars that could have been Bernhard-Anton's.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers N and M, satisfying $1 \leq N \leq 500$ and $1 \leq M \leq 100,000$: the total number of guitars and the number M.
- One line with N different integers $S_1 \ldots S_N$ satisfying $0 \leq S_i \leq 100,000$: the serial numbers of the guitars.

Integers on the same line are separated by single spaces, and there will always be at least one subset of guitars such that the sum of their serial numbers is a multiple of M.

Output

For every test case in the input, the output should contain a single number, on a single line: the maximum number of guitars such that the sum of their serial numbers is a multiple of M.

Example

Input	Output
2 3 5 1 8 6 6 9 8 6 4 1 2 3	3 5

This page intentionally left blank.

D Equal Is Not Really Equal

Problem

Nowadays, many programming languages have standard functions to check if two strings are equal. Even without such standard functions, it is easy to perform such a check: you just have to compare the characters at corresponding positions in both strings.

Sometimes, however, you do not just want a yes-no decision. If two strings are not exactly equal, but the differences are small, then you would like to be informed about that. It is not satisfactory to simply count the number of equal characters at corresponding positions in both strings, and relate this to the length of the strings. For example, strings `"ABCDEFABCDEF"` and `"BCDEFABCDEFA"` are almost equal (they are even equally long), but characters at corresponding positions are all different.

A relatively simple measure of equality that does not suffer from this problem, considers pairs of consecutive characters in the strings. For example, the string `"ABCDEFABCDEF"` contains eleven pairs of consecutive characters: two pairs `"AB"`, two pairs `"BC"`, two pairs `"CD"`, two pairs `"DE"`, two pairs `"EF"` and one pair `"FA"`. Ten of these pairs also occur in the string `"BCDEFABCDEFA"`. Only one pair `"AB"` is exchanged for a pair `"FA"`. Hence, one could say that the measure of equality of the two strings is $\frac{10}{11} \approx 91\%$, which seems to be reasonable. It is not too difficult to extend this measure to strings of differents lengths.

There is, however, a disadvantage. If two strings have the same length and the measure of equality for them is 100%, then this does not necessarily mean that they are equal. For example, both the string `"ABACA"` and the string `"ACABA"` contain one pair `"AC"`, one pair `"CA"`, one pair `"AB"` and one pair `"BA"`. Now, we are curious if this is possible for an arbitrary string. That is, given a string x, does there exist a different string y of the same length, which has measure of equality 100%? If so, then we say that x is *not unique*. Otherwise, we call x *unique*.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with a string x, for which $1 \leq \mathit{Length}(x) \leq 10,000$ and which consists of upper case letters, i.e., characters from the alphabet { A, B, C, ..., Z }.

Output

For every test case in the input, the output should contain a single line, containing `"unique"` if the string x in the input is 'unique', and `"not unique"` otherwise.

Example

Input	Output

`2`
`ABCDEFABCDEF`
`ABACA`
	`unique`
	`not unique`

This page intentionally left blank.

E The Great Cleanup

Problem

It happens to all of us. While you are happily downloading a movie or copying a file, a warning appears on your computer screen: "disk full," or "disk quota exceeded."

There are several ways to deal with this. You may simply accept the fact that you will never watch the movie you were downloading, or that you have to live the rest of your life with a single copy of the file. You may also install a larger disk or try to acquire a larger disk quota.

A third option is to create some space on the disk by removing files that you do not really need anymore. Under Linux, you can use the command `rm` for this. The syntax is simple: `"rm filename"` removes the file with name `"filename"`. You may use a separate rm-command for every single file, but you may also use a wildcard `'*'` to remove multiple files in one step. For example, `"rm BAPC*"` removes all files that start with `"BAPC"`. This way, you have to type fewer commands.

Of course, you must not remove files that you want to keep. Hence, `"rm *"`, which is allowed, and which removes all files in the current directory, is often not desirable.

Now, given the names of the files you want to remove, and given the names of the files you want to keep, you have to determine the minimum number of rm-commands to get the job done. You may only use wildcards *at the end* of your commands. For example, `"rm *.txt"` (which would remove all `".txt"` files) is not allowed.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with an integer N_1, satisfying $1 \leq N_1 \leq 1000$: the number of files that must be removed.
- N_1 lines, each with the name of one file that must be removed.
- One line with an integer N_2, satisfying $0 \leq N_2 \leq 1000$: the number of files that must not be removed.
- N_2 lines, each with the name of one file that must not be removed.

Each filename is a string x with $1 \leq \mathit{Length}(x) \leq 20$, consisting of alphabetic characters (upper case and lower case), digits and/or periods. That is, characters from the set { A, B, C, ..., Z, a, b, c, ..., z, 0, 1, 2, ..., 9, . }. All $N_1 + N_2$ file names for a test case are different.

Output

For every test case in the input, the output should contain a single number, on a single line: the smallest number of rm-commands to remove exactly the right files.

Example

Input

```
1
11
BAPC.in
BAPC.out
BAPC.tex
filter
filename
filenames
clean
cleanup.IN1
cleanup.IN2
cleanup.out
problem.tex
5
BAPC
files
cleanup
cleanup.IN
cleaning
```

Output

```
8
```

F Stock Market

Problem

The bankers of PigsBank, Inc. have collected day-to-day data of the daily profits (and losses) of the shares they hold. Based on these numbers they decide to calculate which days would have been the most profitable to buy and sell their shares, so they can compare that with their actual gain.

Your task is to write a program that computes the optimal 'run' in the sequence of numbers, the subsequence that maximizes the profit. The subsequence you compute is represented by the indexes of the first and last numbers in the subsequence, where we start counting indexes by 1: PigsBank, Inc. bankers are not C programmers. We are asking for exactly one date to buy and one date to sell shares since otherwise the solution would be simple: keep your shares on dates that the profit is non-negative.

Unfortunately that will not help PigsBank, Inc. to avoid the losses they had in the past. Instead they can use it to show future customers how profitable the market could have been if they had been taking the right decisions.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with a single integer N, satisfying $1 \leq N \leq 1,000,000$: the length of the sequence to follow.
- One line with N integers p_i, satisfying $-1,000 \leq p_i \leq 1,000$: the profit (or loss) on date i. The integers are separated by single spaces. At least one of the integers is positive.

Output

For every test case the output contains a single line with two integers i and j, satisfying $1 \leq i \leq j \leq N$, such that the sum of the i-th until the j-th integer is maximized, boundaries included. When i and j have different solutions, choose minimal i and minimal j.

Example

Input	Output
2 11 -3 1 -1 2 3 1 -1 2 -3 -5 7 9 1 -2 3 -1 -1 3 -2 2 -4	2 8 3 6

This page intentionally left blank.

G Acrobat Reader

Problem

At Schiphol airport they use biometrical data to check if people are who they claim to be. When the circus will travel abroad next month, this is expected to give some problems, because when acrobats are passing border control you never know in what orientation their face will be. You can assume they will look straight into the camera, but their face can be rotated by a multiple of 90 degrees. Moreover, as with any passenger, their picture may be translated and scaled (by the same factor in both dimensions).

Given a number of pairs of biometric scans, one from the passport and one freshly recorded, determine for each of these acrobats whether his or her scans match or not.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with an integer N, satisfying $1 \leq N \leq 10,000$: the number of points in each of the two biometric scans for an acrobat.
- N lines, each with two integers x_i and y_i, satisfying $-10,000 \leq x_i, y_i \leq 10,000$: the x- and y-coordinates of a point in the first biometric scan.
- N lines, each with two integers x_i and y_i, satisfying $-10,000 \leq x_i, y_i \leq 10,000$: the x- and y-coordinates of a point in the second biometric scan.

Integers on the same line are separated by a single space. No two points within a single biometric scan are identical. The order of the N points in a biometric scan is completely arbitrary.

Output

For every test case in the input, the output should contain a single string, on a single line: `"okay"` if the scans match, or `"mismatch!"` if they do not.

Example

Input

```
2
3
-1 1
0 -1
1 0
-1 0
1 -2
3 2
3
0 0
2 1
2 2
0 0
-2 1
-2 2
```

Output

```
okay
mismatch!
```

This page intentionally left blank.

H Farmer John

Problem

Farmer John owns a lot of cows that graze in the fields and walk around happily. However, cow Bessie is very lazy and always takes the shortest route to the barn to get food. Farmer John wants to give Bessie some more walking exercises, so he placed some extra fences to make sure that Bessie cannot always take the shortest route and has to walk around the fences.

Given the current location of Bessie, the location of the barn with the food, and all locations of the fences (modelled as line segments), farmer John wants you to compute the minimum distance that Bessie has to walk. Bessie is not allowed to cross any fence on her route, but she is allowed to touch the fences.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with four integers B_x, B_y, F_x and F_y satisfying $-10,000 \leq B_x, B_y, F_x, F_y \leq 10,000$: the location of Bessie and the location of the food.
- One line with one integer N satisfying $0 \leq N \leq 100$: the number of fences.
- N lines, one for each fence, with four integers x_1, y_1, x_2 and y_2 satisfying $-10,000 \leq x_1, y_1, x_2, y_2 \leq 10,000$ and $x_1 \neq x_2$ or $y_1 \neq y_2$: the x- and y-coordinates of the begin and end points of this fence.

Integers on the same line are separated by single spaces. The current location of Bessie and the location of the food will not lie on a fence, and fences will not touch or overlap.

Output

For every test case in the input, the output should contain a single real number, rounded and displayed to six digits after the decimal point, on a single line: the minimum walking distance.

Example

Input	Output
3 0 0 10 0 1 -1 -100 -1 100 0 0 10 0 2 -1 -100 -1 100 5 4 5 -6 0 0 2 1 1 1 1 1 -1	10.000000 12.806248 2.414214

This page intentionally left blank.

I Imagine

Problem

In some imaginary city made up especially for this problem, there are two fictional political parties that were named A and B (because these names are conveniently short and also nobody really cared).

In the centre of this non-existent city you are supposed to imagine a large billboard of 10.24 metres in width and 10.24 metres in height, essentially a 1024×1024 grid of square centimetres. Every now and then a political activist comes around and puts a 1cm $\times$ 1cm sticker with either an A or a B, depending on his or her assumed political orientation, on the grid. Any sticker that was put on that particular position before, no matter from which of the parties, becomes invisible. This might all sound a wee bit unrealistic, but you need to keep in mind that it is in fact not real. It is just for the problem.

Now, for some obscure reason, that could have been totally justified had someone put a bit more effort into writing this assignment, a list is maintained with the exact location and denomination of every single sticker that was put up, in chronological order.

At any moment one might wonder how many A's and B's there currently are within some subrectangle of the grid. Such question may come up, for example, when someone looks at the billboard from a large distance, with a rectangular telescope. You will find such queries in the input, mixed with the information about when what kind of sticker was put where, all in chronological order. Can you write a program that can handle such queries quickly?

You may assume that the grid has been initialized like a checker board with an A in the top left corner at $x = y = 1$, hence in the following way:

A	B	A	B	A	B	A	B	A	
B	A	B	A	B	A	B	A	B	
A	B	A	B	A	B	A	B	A	
B	A	B	A	B	A	B	A	B	

Input

The input consists of a single test case, which has the following format:

- One line with an integer N, satisfying $1 \leq N \leq 1,000,000$: the total number of stickers and queries in the input.
- N lines, each of which is formatted as one of two alternatives:
  - A line starting with the character `'A'` or `'B'` followed by two integers x and y, satisfying $1 \leq x, y \leq 1024$, denoting a sticker which is put on the grid.
  - A line with the character `'R'` followed by four integers x_1, y_1, x_2 and y_2, satisfying $1 \leq x_1 \leq x_2 \leq 1024$ and $1 \leq y_1 \leq y_2 \leq 1024$, denoting the top left and bottom right coordinates of a queried subrectangle.

Alphabetic characters and integers on the same line are separated by single spaces. You can make no assumptions about the way the stickers and queries are interleaved in the input.

Output

For every query in the input, the output should contain two numbers, on a single line: the numbers of A's and B's in the queried subrectangle at the moment of the query, separated by a single space.

Example

Input

```
7
R 1 1 3 3
A 2 3
A 1 3
R 1 1 3 3
B 2 2
B 3 3
R 2 2 3 3
```

Output

```
5 4
6 3
1 3
```

J My Cousin Obama

Problem

When Barack Obama was elected president of the United States of America, the city of Leiden was proud to announce that his ancestors Thomas Blossom and Anne Eldson had lived in Leiden between 1609 and 1629. They were members of over 125 protestants which had fled from England to Holland, seeking religious freedom, and which later became known as the 'Pilgrim Fathers'. Father and son Bush (earlier presidents of the USA) were descendants from the same Leiden couple.

Of course, we wish Leiden joy of having a relation with Barack Obama, but Leiden is probably not the only city in the world with such a relation. Given that every person has two (biological) parents, and that you get a new generation every, say, thirty years, Barack Obama probably has several thousands of ancestors from the early seventeenth century.

It would be more interesting if the relation of Obama with Leiden would be entirely along the male line of ancestry.[1] This is not the case: Obama's father is from Kenia, and in the fact, the line towards Thomas Blossom and Anne Eldson started from Obama's mother.

In general, when some person A_0 descends from some person B_0 living centuries ago, there is not necessarily a unique line of ancestry. Far relatives may get married without even knowing that they are relatives. In such a case, it would be interesting to find the line of ancestry from A_0 to B_0 containing as few women as possible.

You are asked to find this line of ancestry, from a database with ancestor information. In particular, you are asked for the smallest *number* of women on a line of ancestry from A_0 to B_0. For the sake of simplicity, the database does not contain real names, but numbers between 1 and a certain maximum N to identify individuals.

Input

The first line of the input contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with one integer N, satisfying $2 \leq N \leq 100,000$: the number of persons in the database.
- One line with two different integers A_0 and B_0, satisfying $1 \leq A_0, B_0 \leq N$: the ID numbers of the two persons we are interested in.
- N lines, each with two integers f and m, satisfying $0 \leq f, m \leq N$. The integers on line i are the ID numbers of the father (f) and the mother (m) of the person with ID number i. An ID number $f = 0$ (or $m = 0$) indicates that the father (or mother) of person i is not known.

Integers on the same line are separated by single spaces.

The integers f and m on the N lines are gender-consistent: no person is registered both as father and as mother (of the same or different persons).[2] We do not assume age limits; for example, a man and a woman who are ten generations apart, may have a child together. Of course, the ancestor information in the input does not describe cycles.

Output

For every test case in the input, the output should contain a single line, containing the smallest number of women on a line of ancestry from A_0 to B_0 (not counting A_0 and B_0 if either of them is a woman) if B_0 is an ancestor of A_0 according to the database, or containing the string `"no ancestor"` otherwise.

[1] We apologize, if people feel offended by this traditional view on the role of men and women in a line of ancestry.
[2] We apologize if people feel offended by this traditional view on the sex of a person.

Example

Input

```
2
23
1 8
2 0
3 15
9 4
5 10
11 6
0 7
8 0
0 0
0 0
0 0
12 14
13 0
0 0
0 0
16 23
17 20
0 18
19 0
12 7
21 0
22 0
8 0
0 0
9
2 9
5 2
3 4
0 0
0 0
6 7
0 0
8 9
0 0
0 0
```

Output

```
2
no ancestor
```