North Western European Championship Programming

ACM's North Western European Regional Collegiate Programming Contest

1997/1998

Problem A :	Arrows
Problem B :	MDST- machine
Problem C :	Entropy
Problem D :	Columns
Problem E :	Internet
Problem F :	Gardener
Problem G :	Mendel
Problem H :	Hall of Flames

Department of applied mathematics and computer science

November, 23rd, 1997

Problem A : Arrows

inputfile: input.A
outputfile: output.A

Description

The game 'Arrows' is played on a board, on which a coloured directed graph is drawn. Not only the locations in the graph are coloured, also the arrows pointing from location to location are coloured. Each of the two players has a piece. At the beginning of the game, the pieces of both players are in fixed positions on the board. In turn, the players may do a move. A move consists of moving one's own piece along an arrow to a new location on the board. The following constraint is imposed on this: the piece may only be moved along arrows of the same colour as the location of the opponent's location. For example, if the opponent's piece is currently at a red location, the player may only move his own piece along a red arrow.

In the sixties ('*make love not war*') a one-person variant of the game emerged. In this variant one person moves both pieces, not necessarily one after the other. Goal of this game is to get both pieces onto the same location, using as few moves as possible.

Problem

The problem to be solved is to find out the smallest number of moves that is necessary to get both pieces into the same location, for a given board layout and starting positions.

The board locations are identified by upper-case letters ranging from A to Z. The colours of the locations and arrows are represented by the letters a to z.

Input

The input file starts with a line containing the number of games to be analysed.
Each of the game descriptions starts with a line containing the number of locations on the board. For each location, there is one line describing it.
This line first contains an upper-case letter denoting the name of the location, followed by a lower-case letter denoting its colour. The number of arrows leaving this location follows. Then for each of these arrows, there are two letters, first an upper-case letter denoting the endpoint of the arrow, followed by a lower-case letter denoting its colour. All fields are separated by a single space.

There are no arrows from a location to itself, and for every pair of locations, there are at most two arrows between them (at most one in each direction).

Finally, for each game description, there is one line containing two upper-case letters, which are the starting positions of the game pieces.

Output

The output file should contain one line for every analysed game, containing the minimum number of moves required to get both pieces into the same location, or the word `impossible` if that is not possible for the given board and starting position.

Example

Sample input

```
3
4
A r 1 B y
B g 1 A b
C b 1 A r
D y 1 C g
A D
3
A x 1 M x
Z x 1 M x
M x 0
A Z
2
A x 1 M y
Z x 1 M y
M x 0
A Z
```

Correct output for the sample input

```
4
2
impossible
```

Problem B : MDST- machine

inputfile: input.B
outputfile: output.B

Description

The object, later known as the MDST-machine, was discovered in 1991 nearby West-Knollendam. The object was made of an unknown metal. The surface was solid, smooth and seamless. All the attempts to open the object failed miserably.

On the outside only a couple of buttons and windows were visible. The button to activate the object could be recognised easily, but it didn't seem wise to activate the object without knowing its functionality. A special council, initiated by the United Nations, decided to activate the object in a specially secured environment.

D-day. The whole world watched what would happen. After a long countdown finally the object was activated... Nothing seemed to happen. No sound could be heard. Suddenly two displays lightened. The first display could be used to enter series of operations. The second display could display 9-digit numbers.

In the years that followed scientists worked hard to discover its functionality. The object appeared to be capable of manipulating numbers. For that purpose four operations exist.
These operations are:

- `mul2` (abbrev. `M`): Multiplies a given number by two
  e.g. mul2 3051 = 6102
- `div2` (abbrev `D`): Divides a given number by two
  e.g. div2 3051 = 1525
- `sort` (abbrev. `S`): Sorts the digits of a number in ascending order. Leading zeros disappear.
  e.g. sort 3051 = 135
- `tros` (abbrev. `T`): Sorts the digits of a number in descending order (opposite of sort)
  e.g. tros 3051 = 5310

Series of operations can be entered, which are successively applied to a given number. The series is processed from left to right.
e.g. MSDT 3051 = SDT 6102 = DT 126 = T 63 = 63

The functionality of the object was revealed. Still it wasn't clear for whatever purpose the object was used and by whom.
In a small cave in Peru inscriptions were found which might lead to the final answer. These inscriptions form a special number-magic. For several numbers a series of operations was given which finally lead to the number 1.

The next relations are found:

- A given representation is the shortest representation possible
- If there is more than one 'shortest' representation, the alphabetically shortest representation is given
- Only the second display is used to display the resulting numbers. So after any operation the resulting number cannot be larger than 9 digits. If an operation results in a number larger than these 9 digits, the first part of this number is ignored and only the last 9 digits are used for further operations.

Problem

Given a number n *(1 ≤ n < 20000)* calculate the smallest series of operations leading to 1. If more than one series is applicable, the alphabetically smallest has to be chosen.

Input

The input file start with a line containing the number of tests t in the inputfile, followed by t lines, each containing a single number, of which the smallest series leading to 1 has to be calculated.

Output

For each corresponding test in the input file a single line is produced, containing the smallest series leading to 1. If more than one series is applicable, the alphabetically smallest is given.

Example

Sample input

```
7
7
25
30
51
100
137
537
```

Correct output for the sample input

```
DD
MMS
SD
DMMS
S
DDTDDS
TMMSDDSD
```

Problem C : Entropy

inputfile: input.C
outputfile: output.C

Description

After her brilliant discovery of worm holes exactly one year ago, the physicist returns to her daily work, which consists mainly of doing experiments and analysing the measurement data. Because of the low funding of her research group she has to use simple devices like a cheap Entr-o-Meter™ to measure the entropy of the universe. Despite the second law of thermodynamics, which states that entropy must always be increasing, her measurement data does not exhibit that pattern. She suspects that the cheap Entr-o-Meter has a loose connection and that only some of the values in the measurement data actually are entropy values, and the rest is just noise. She needs your help to extract the entropy data from her measurements. Buying a more expensive Analyz-o-Matic® is not an option.

Problem

The measurements consist of sequences of seemingly random numbers. The first step in the analysis, for which your help is needed, is to determine the number of valid data elements in each of these sequences. The physicist's hypothesis is that, if you take a selection of the measurement values, the longest strictly increasing selection is the 'real' entropy data. For example, if the measurements are `6 4 5 5 4 10`, the hypothesis says that the entropy values are `4 5 10`. Of course, this is still a hypothesis. To test the hypothesis, the physicist wants you to write a program that determines for each of the measurement sequences the *length* of such a selection.

Input

The input file starts with a line containing the number of measurement sequences to be analysed.

Each measurement sequence starts with a line containing the number of measurement values n ($1 \le n \le 1000$), followed by a line containing the actual measurement values, integers between -1000000 and 1000000, separated by single spaces.

Output

The output file should contain one line for each measurement sequence, containing the length of the longest strictly increasing selection of measurement values from that sequence.

Example

Sample input

```
4
6
6 4 5 5 4 10
3
0 -1000000 0
6
1 1 2 2 3 3
6
3 3 2 2 1 1
```

Correct output for the sample input

```
3
2
3
1
```

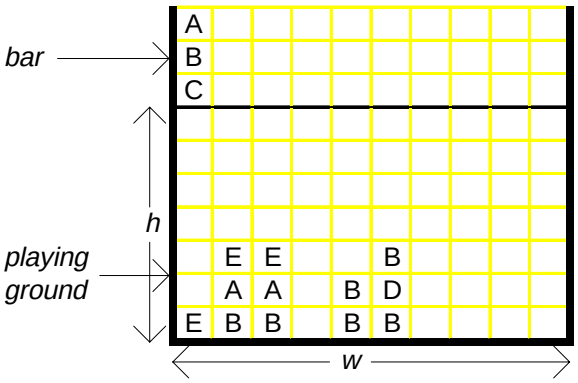
Problem D : Columns

inputfile: input.D
outputfile: output.D

Description

The game *Columns* is rather well known, at least in its dynamic form. The player can manipulate a bar of coloured squares while it drops down. Once it has reached the playing field, points are gained and squares in the playing field disappear, according to certain rules. This game is very exciting, as the bars will drop down faster and faster as the game goes on. For some people it turns out to be too exciting, as they get a nervous breakdown from playing this game. For those people the static variant was invented.
(Of course not playing at all would be a better idea. But the real addicts get a nervous breakdown if they don't play as well.)

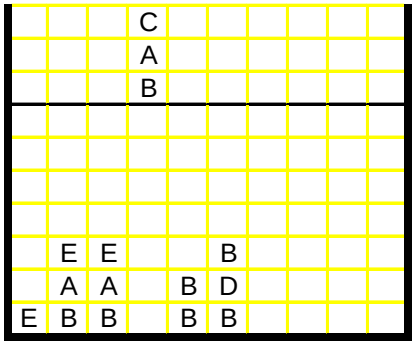
Let me explain *Static Columns* to you. The playing field is a rectangle of width w and height h. Initially it is empty, but after some time it may be filled with coloured squares. In the scheme below the colours are indicated with capitals. The squares are subject to gravity: a square rests either on the floor or on another square. Repeatedly a vertical bar (width 1) of coloured squares appears on top of the playing field, against the left wall.



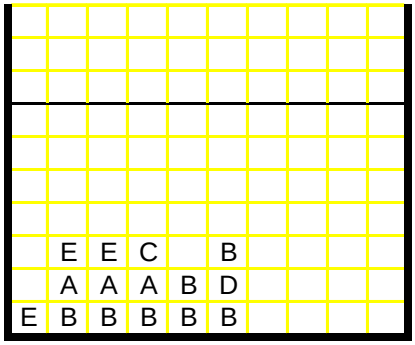
The player may manipulate this bar using (repeatedly) any of the following commands:

Command	Action
r (right)	Move the bar one step to the right. This command has no effect if the bar is already at the rightmost position of the playing field.
l (left)	Move the bar one step to the left. This command has no effect if the bar is already at the leftmost position of the playing field.
s (shift)	Shift (or actually rotate) the colours in the bar. The rotation is downwards: every colour goes down one step, and the lowermost colour is put on top.
d (drop)	Drop the bar. After this command, no other commands can be given for that bar.

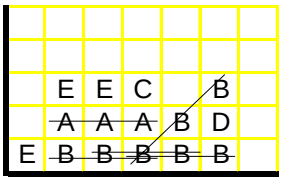
After the commands `rrrs` the situation in the above example will be:



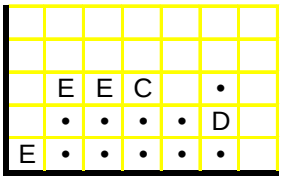
The drop command makes the bar going straight down until it is stopped, either by the floor or by a coloured square.



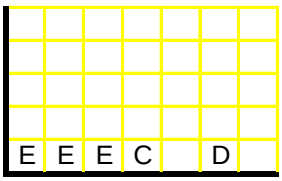
If the playing ground is very full, or the bar is very long it may happen that the bar is stopped before it is fully inside the playing ground. In that case the game is over and no other bars will be played. If not, two things happen. First, credits are counted, depending of the parameter l, the length of a series. Whenever l squares in the field in a series (horizontal, vertical or diagonal) have the same colour, one point is given to the player. As indicated in the next diagram, the player receives 5 points if $l = 3$ (the player would receive 2 points if $l = 4$ and 1 point if $l = 5$).



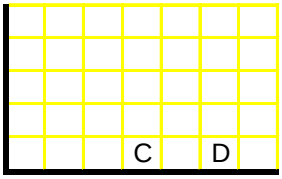
After all squares occurring in one or more such a series have disappeared, the following diagram is obtained:



Now gravity takes over, and the diagram becomes:



Once again a series is found and the player obtains one more point. The final situation is:



Problem

The *World Championships Static Columns* will be held next year in a very obscure place slightly North of nowhere. Because this place is far away from everywhere, not many people will actually go there, but lots of people are anxious to know what is going on there. Using the Internet every move of every player is broadcasted all over the world. People with beautiful graphics on their machine will be able to replay the game as if they were actually there. Others are less well supplied, and are happy if they know the credits obtained by each player. You are asked to write a program to calculate the credits.

Input

The first line of input contains the number of games g. Then g games follow.

Every game starts with a line with 4 numbers, the parameters of the game. The first number is the width w of the board: $(1 \leq w \leq 500)$. The next number is the height h of the board: $(1 \leq h \leq 250)$, The third number is the length l of a series: $(1 < l < 10)$. The last number is the number of moves m to follow. Then m moves are given, each on a line.

A move is given by a sequence of characters. It starts with one or more capitals, indicating the colours of the bar, from top to bottom, followed by the commands of the player, represented by the letters `l` (left), `r` (right) and `s` (shift). A move always ends with the letter `d` (drop). A move is never longer than *5000* characters.

Output

The output consists of g lines, one line for each game. In case of game over this line consists of the text `GAME OVER`, followed by a space and the credits gained until the game was over. (Commands after a `GAME OVER` are to be ignored). If all the bars are dropped and there was no `GAME OVER`, the credits gained so far is to be given.

Example

Sample input

```
2
6 7 3 6
Ed
EABrd
EABrrd
BBrrrrd
BDBrrrrrd
ABCrrrsd
2 4 3 2
ABCllrrd
ABCrrrrd
```

Correct output for the sample input

```
6
GAME OVER 0
```

Problem E : Internet

inputfile: input.E
outputfile: output.E

Description

As most of you know, the Internet is a global network connecting thousands and thousands computers world-wide. It is also well known that the 'net' is a packet-switched network. This means that it consists of computers with unidirectional point-to-point links between them. If a host receives a communication packet that is not addressed to it, it will forward it along one of its outgoing links. This allows communication between computers that are not directly connected to each other.

Problem

Originally, one of the most important goals of the 'net' was to provide communications capability even after parts of the network were destroyed by e.g. nuclear attacks. In order to verify whether this is really true, we want you to write a program that checks whether communication between all computers on the network is still possible after something bad has happened.

Input

The input starts with a line consisting of one number describing the number of scenarios to be analysed.
Each scenario starts with a line consisting of one number describing the number of communication links in this scenario, followed by a line for each communication link. These lines consist of two hostnames, separated by a single space, indicating that the first host can transmit packets to the second host. A hostname consists of at least 1 and not more than 8 lower case letters. A communication link is never from one computer to itself. There are at most 100 computers in the network.

Output

The output should contain one line for each scenario, consisting of the single word `yes`, if communication between all computers is possible, and `no` if this is not possible.

Example

Sample input

```
2
2
mit tudelft
tudelft mit
3
pi wxs
wxs pi
xsall wxs
```

Correct output for the sample input

```
yes
no
```

Problem F : Gardener

inputfile: input.F
outputfile: output.F

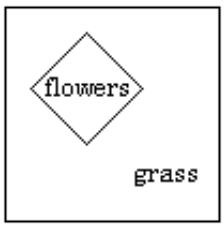
Description

Like computer programming, gardening isn't just a hobby---it's a science. Much more so if you have to design the layout of a brand new garden according to several rather mysterious boundary conditions.

This is just the problem faced by gardener Jeeve Stobs[1]. Just retiring from the computer business (which '*isn't what it used to be*', according to Mr. Stobs), he now tries to make a living out of designing gardens according to the whims and wishes of the rich and famous. His latest employer, a certain Mr. G. Bates, has given him the following requirements for the garden next to his newly built house:

- the garden is to be square, with each side measuring an integer number of feet.
- the garden must have a fence around it.
- Mr. Bates is very keen on grass. An area of N square feet (N also being an integer) will have to be covered with grass.
- Within the garden, there will have to be a square-shaped flower-bed, measuring an positive integer number of feet on each side.

Thus, a garden will look something like this:



Sometimes, it is impossible to meet the boundary conditions mentioned above. Most of the time, however, it can be done, and in those cases Mr. Stobs decides he wants to maximise his profit per square foot of garden. His profit is 0.80 dollar per square foot of flowers and 0.30 dollar per square foot of grass. The fence, which he can get very cheaply (but Mr. Bates doesn't have to know that!) earns him a profit of 80 dollar per foot.

Problem

The question is this: given N, what is the maximum profit per square foot of the garden Jeeve will design, meeting the boundary conditions mentioned above.

[1] any resemblance to actual people is purely accidental/coincidental (or isn't it?)

Input

The first line of the input contains the number of trials T ($0 \leq T \leq 10000$). After that T lines follow, each containing a number N: the area of garden which is to be covered with grass. This area is a positive integer number, less than or equal to 2000000000.

Output

For each trial a single line has to be given in the outputfile with the profit per square foot of Jeeves garden-design (exactly 4 digits precision following the floating point). If it is impossible to meet the boundary conditions, the word `impossible` has to be given.

Example

Sample input

```
8
10
100
1000
10000
100000
1000000
10000000
100000000
```

Correct output for the sample input

```
impossible
13.0337
9.5347
3.0400
1.3112
0.8512
0.8051
0.8005
```

Problem G : Mendel

inputfile: input.G
outputfile: output.G

Description

Two months ago one of the judges became father of a son. Before the child was born a lot of speculation was going on about its appearance. You should know that both parents have a Chinese father (dark hair, brown eyes) and a Dutch mother (blond hair, blue eyes). Although both parents have an Oriental look, there was a reasonable chance the baby would have blue eyes and blond hair. The final outcome was less surprising. The Oriental influences were clearly visible when the baby-boy was born. This is not his story. Some feared it would be politically incorrect. This one is about another planet.

On Koseb, a few light years away, live two kinds of beings: Jona and Nathan. Each kind has its own characteristics but the most striking difference is the colour of their left ear. A Jona-being has a green left ear, a Nathan-being has a purple left ear.
Both species lived separatedly for many years, but increasing global mobility has led to many mixed engagements. The first descendants always had a purple ear, but the colour of the left ear of their descendants couldn't be predicted again.

After long research scientists found out that each being has exactly two genes that determine the colour of the left ear. Only three combinations exist:

- pure: `Green Green` or `Purple Purple`
- mixed: `Green Purple` (is the same as `Purple Green`)

It appeared that Purple was dominant over Green, so someone with a mixed combination always has a purple left ear.
When two beings get a descendant, the descendant gets one single gene of each parent. This gene is randomly chosen. For example if one of the parent has a mixed combination (GP) and the other has a pure green combination (GG), then the descendant has an even chance to get a pure combination (GG), a green left ear, or a mixed combination (GP), a purple left ear.

When the beings on Koseb finally found out what caused the colour of their ears, they still couldn't predict colours, because the gene-combinations of the ancestors were unknown. Fortunately all the past engagements were well administrated, going back for many years, even to the time that there were no mixed engagements.

Problem

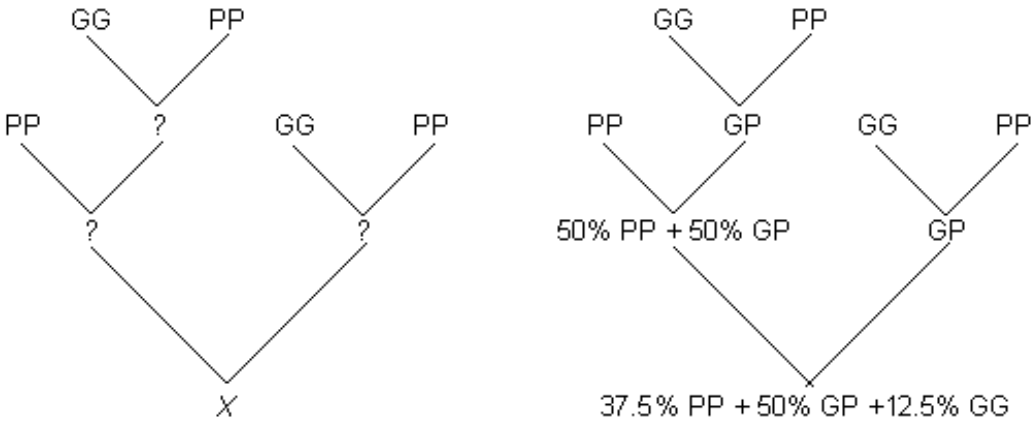
Build a program that calculates the chances for the colour of the left ear of a certain person if the family trees of both parents are known. The leaves of a family tree are always pure combinations.

Example

The chances for the being at *X* can be calculated as followed:

- The engagement between GG and PP results in a 100% GP combination.
- The engagement between PP and GP(100%) results in a GP(50%) or a PP(50%) combination.

- The GP(50%)+PP(50%) combination and the GP(100%) combination result in: PP(37,5%)+GP(50%)+GG(12,5%) combination.



So the being at *X* has a 87.5% chance to have a purple left ear and a 12.5% chance to have a green left ear.

Input

The inputfile start with a single line, which contains the number of testruns t in the file. A testrun consists of two lines. Each line indicating the family tree of one of the parents.
In the family tree the gene for a purple left ear is indicated with the character 'P' and the gene for a green left ear is indicated with the character 'G'.
In a family tree a being is defined by its genes or its two parents.

```
genes  :: PP | GG
being  :: genes | (parent-being,parent-being)
```

In the example the left family tree is defined as: (PP,(GG,PP)) and the right family tree is defined as (GG,PP).

Output

For each testrun from the input one single line has to be generated with the chance that the being has a purple left ear. The chance has to be given in percentages in one decimal precise.

Sample

Sample input

```
2
(PP,(GG,PP))
(GG,PP)
(PP,GG)
(GG,PP)
```

Correct output for the sample input

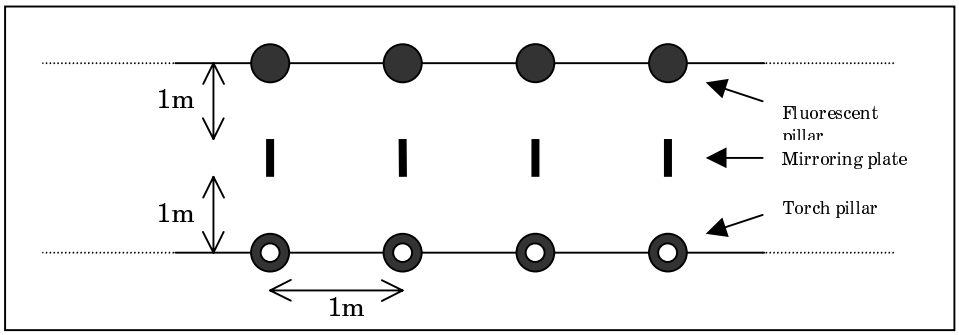
```
87.5
75.0
```

Problem H : Hall of Flames

inputfile: input.H
outputfile: output.H

Description

Recent archaeological work has exposed an artefact that could well be the largest piece of art from Roman history ever found. It is a large gallery formed by 100 pillars at each side, one meter apart. The gallery itself has a width of 3 meters. Although the ravages of time have demolished quite a bit, the site clearly shows that the pillars on one side were equipped with a torch, where the pillars on the other side were painted with a fluorescent paint (another discovery: such a paint was never found before).
In the middle between two pillars at either side, some kind of metal plate (width: 1 meter) with high mirroring capacity was attached. This is a schematic drawing of the artefact:



Problem

The researchers think that the artefact played a role in the evening spectacles of the old Romans. By lighting or extinguishing torches and removing mirrors, the appearance of the fluorescent pillars could be changed. Although the artefact is in such a state that a lot of research could be done, the effect cannot be shown anymore. That is where you come in. You should write a program that, given the places of torches that are lit and the places of mirrors that are present, prints out those places of fluorescent pillars that will not get illuminated, either directly by a torch, or indirectly through mirrors. The places are numbered from left to right starting from *1*.
The researchers found out that a mirroring plate mirrors on both sides, but the endpoints absorb the light (this also means that the light from the flame directly opposite to the mirror is absorbed). The dimensions allow for considering the pillars (& torches) and mirrors as points and lines in an integer grid. A pillar will light up when it is hit by any intensity of light.

Input

The input file starts with the number of cases. Each case exist of:

- A line with the number of lit torches T $(0 \le T \le 100)$
- A line with the places of lit torches
- A line with the number of present mirrors M $(0 \le M \le 100)$
- A line with the places of the present mirrors

Output

Per case, the output file should contain one line with the places of pillars that **don't** get illuminated, either directly or indirectly. These numbers should be presented in ascending order, separated by a single space. No trailing spaces allowed.

Example

Sample input

```
2
1
20
2
19 21
1
5
1
5
```

Correct output for the sample input

```
17 18 22 23
5
```