BENELUX
ALGORITHM
PROGRAMMING
CONTEST

2010 LEIDEN

Powered by Calco-IT

23 October 2010

Problems

Problems BAPC 2010

And the colours of the corresponding balloons.

A	–	Gene Shuffle	White
B	–	Top 2000	Dark Green
C	–	The Twin Tower	Lemon Green
D	–	Collatz	Orange
E	–	Clocks	Purple
F	–	Maze Recognition	Black
G	–	Snooker	Dark Blue
H	–	Pie Division	Light Blue
I	–	Keylogger	Red
J	–	Wrong Answer	Yellow

Large Output

For some of the problems, the output may be large. In particular, it may consist of many components (characters, numbers). If you program in Java and you write these components one by one, this may result in a Time Limit Exceeded.

There are probably many ways to avoid this. One way is to use a BufferedOutputStream, as in the following example.

```
import java.io.*;

PrintStream out = new PrintStream(new BufferedOutputStream(System.out));

char a;
out.print(a);
int b;
out.println(b);
...

out.close();
```

After you have created the `PrintStream out` in the second line, you can use it just like you would use `System.out`. Do not forget the instruction `out.close();` at the end. It is necessary to (eventually) write all output to standard output.

A Gene Shuffle

Problem

The genomes of two related species, like that of cabbage and turnip, may contain similar genes. Their order in the genome may be different due to genome transpositions during evolution.

Your task is to compare two gene sequences, and to determine the segments that are common to the genomes, i.e., the segments that contain the same genes in both genomes (although in a possibly different order).

Genes in this application are not given by sequences of bases A, C, G and T as is usually done, but are coded as a single integer. The two genomes that are compared consist of the same set of integers, each a permutation of the numbers $1, \ldots, N$, where N is the length of the genome. A common segment i-j of the genomes is an interval $1 \leq i \leq j \leq N$ such that elements starting at position i and ending at position j of the first genome make up the same set of genes (integers) as the elements at the same positions in the second genome, but possibly in a different order. We are looking for *minimal* common segments, i.e., segments cannot contain smaller segments.

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 $1 \leq N \leq 100,000$: the length of the genome.
- One line with a permutation of the N integers $1, \ldots, N$, representing the first genome.
- One line with a permutation of the N integers $1, \ldots, N$, representing the second genome.

Integers on the same line are separated by single spaces.

Output

For every test case in the input, the output should contain a single line, listing the minimal segments i-j common to the two genomes, ordered from left to right. Segments are separated by single spaces.

Example

Input

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

Output

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

This page intentionally left blank.

B Top 2000

Problem

When the year 2000 was approaching, the Dutch radio station Radio 2 launched the Top 2000: a chart containing the 2000 most popular singles of all times. The chart was based on votes, cast by the listeners of Radio 2. It was broadcasted in the last week of 1999.

Because of the enormous success of the Top 2000, the radio station decided to compile and broadcast a new list every year. This implied in particular that every year, the Radio 2 program makers had to solve a large and complicated scheduling problem. The first few years, they did this by hand. Since the problem came back every year, they decided it would be worth your while hiring a smart student to write a computer program for it.

So what is the problem? Once the votes of the listeners have been cast, the 2000 singles in this year's Top 2000 are known, and so is the order in which they have to be played: from number 2000 (the least popular) to number 1 (the most popular). Each single has a known length, so one can easily calculate how much time one would need to play all 2000 singles, *provided that one can broadcast continually for a sufficiently long time.*

Unfortunately, every hour, there is a five-minute break for commercials and the latest news. Hence, the 2000 singles must be scheduled over blocks of 55 minutes. No single is played (partly or as a whole) more than once. In particular, no single is split over different blocks.

It is unlikely that the singles (with their known lengths) that have to be played in one block, require exactly 55 minutes. If together they are longer, then they cannot be played completely. The only way to deal with this, is to cut some parts of the singles. If, on the other hand, together they are shorter than 55 minutes, the remaining time must be filled with (spoken) information by a DJ.

Neither of these 'solutions' is really appreciated, because the listeners of Radio 2 like to hear complete songs, and do not like to hear the (usually boring) talk of a DJ. In particular, there is a penalty for every minute of a song that is cut, and there is a penalty for every minute that is filled with the talk of a DJ. The task is to schedule the 2000 singles over blocks of 55 minutes, such that the total penalty is minimal.

The number of blocks is not fixed. However, it must be an integer number, since it is common practice that a radio program fills a number of *complete* blocks. For simplicity, we assume that the length of a single is an (integer) number of minutes. At least one second of every single must be played.

Finally, if the Radio 2 manager is happy with the performance of the computer program for the Top 2000, he considers using (or selling) it for other single charts, as well. Therefore, the number of entries (2000 in the Top 2000) and the block size (55 minutes on Radio 2) must not be fixed, but must be read from the input.

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 separated by a single space, satisfying $1 \leq N \leq 50,000$ and $15 \leq M \leq 100$: the number of singles to be played, and the number of minutes per block, respectively.
- One line with two integers A and B separated by a single space, satisfying $1 \leq A, B \leq 1,000$: the penalty for every minute of music that is cut, and the penalty for every minute that is filled with (spoken) information by a DJ, respectively.
- One line with N integers, separated by single spaces: the lengths (in minutes) of the N singles in the chart, in the order in which they have to be played. Each integer x on this line satisfies $1 \leq x \leq 20$.

Output

For every test case in the input, the output should contain a single number, on a single line: the minimum (total) penalty of a schedule for the N singles in the input.

Example

Input

```
3
10 25
2 1
8 7 3 5 4 2 9 4 3 4
16 55
4 1
14 9 13 13 6 15 7 8 13 7 5 11 10 11 9 14
15 28
1 2
7 9 7 5 8 7 6 10 5 9 7 9 6 10 5
```

Output

```
4
0
19
```

C The Twin Tower

Problem

In recent years so many twins enrolled at Leiden University, that housing them has become a big problem. In an effort to accomodate everyone, the university has made plans to build a skyscraper of N floors high, with 9 rooms on each floor, laid out in a 3×3 square. According to these plans, everyone should be able to get a room next to or directly above or below their twin brother or sister. To be even more precise: the two rooms of a twin should either be at opposite sides of a common wall, or the floor of one room should be the ceiling of the other. For the sake of privacy, students never share a room.

Today the university has tasked some poor soul (you!) to count all possible pairings of rooms that leave no room unpaired, modulo $10,007$.

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 $0 \leq N \leq 5,000$.

Output

For every test case in the input, the output should contain a single number, on a single line: the number of possible configurations modulo $10,007$.

Example

Input	Output

`4`
`2`
`4`
`1576`
`2680`
	`229`
	`7728`
	`229`
	`7728`

This page intentionally left blank.

D Collatz

Problem

In the process to solve the *Collatz conjecture*, better known as the $3n + 1$ *problem*, Carl created a physical model with wood and ropes. A wooden bar contains a hole for every natural number from 1 to infinity from left to right. For every even number m there is a rope connecting the mth hole with hole $\frac{m}{2}$. For every odd number n there is a rope connecting the nth hole with hole $3n + 1$.

For an important conference where Carl plans to elaborate on his results, he wants to bring his structure, but it is too large to fit in his bag. So he decided to saw off the part of the bar containing the first N holes only. How many ropes will he need to cut?

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 $0 \leq N \leq 10^9$.

Output

For every test case in the input, the output should contain a single number, on a single line: the number of ropes that need to be cut.

Example

Input	Output
3	10
12	200
240	3000
3600	

This page intentionally left blank.

E Clocks

Problem

During the past few years Tim E. has been collecting circle-shaped clocks and he exposed them on a big wall in his living room. He wants to buy another big clock, but because his wall is almost full he wants to know what the radius of the biggest clock is that could be exposed on the wall without overlapping any other clock.

The wall is modelled as a rectangle with width W and height H. For each clock the coordinates (x_i, y_i) of the centre-point and the radius r_i are given. The clocks will not overlap with the boundaries of the wall, nor will two clocks overlap, however they might touch each other.

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 W and H satisfying $1 \le W, H \le 1,000,000$: the width and the height of the wall, respectively.
- One line with an integer C satisfying $0 \le C \le 50$: the number of clocks.
- C lines, each with three integers x_i, y_i and r_i satisfying $r_i > 0$, $0 \le x_i - r_i$, $x_i + r_i \le W$, $0 \le y_i - r_i$ and $y_i + r_i \le H$: the location and radius of an existing clock.

Integers on the same line are separated by single spaces. For any pair of clocks i and j $(i \neq j)$, $(x_i - x_j)^2 + (y_i - y_j)^2 \ge (r_i + r_j)^2$.

Output

For every test case in the input, the output should contain a single real number, on a single line: the maximum radius of a clock that fits on the wall without overlapping with any existing clock or the boundaries of the wall.

Your answer should have either an absolute or a relative error of at most 10^{-6}. Make sure that you print enough decimals. In particular, if you program in C++, do not use plain `cout << M << endl;` to output your answer (a double) M, because this sometimes produces too few decimals. Instead, you may use `printf ("%lf\n", M);` .

Example

Input	Output

1
10 10
4
2 2 1
2 8 1
8 2 1
8 8 1
	3.242640687119285

This page intentionally left blank.

F Maze Recognition

Problem

You are trapped in a maze that consists of square rooms with doors between some of the adjacent rooms. You do not have a map of the maze. The only things that you know are that there is exactly one exit room (which may be anywhere in the maze), and that the size of the maze will never exceed 100×100 rooms. When you are in a room, you can see in which directions there are doors. If you are in the exit room, you can also clearly see this. You are not only interested in exiting the maze, but you also want to know what the *minimum* number of rooms is that you need to enter before reaching the exit room. Write a program that searches the maze for you, and decides what the shortest possible route would have been.

Input and output

This is a problem with dynamic input and output, read the following description very carefully.

The first line of the input contains a single number: the number of test cases to follow. Then, for every test case the input will depend on the values that your program outputs. The sequence of input and output is as follows:

1. Input: One line with a string containing the possible moves that you can make in the current room. These moves can be any subset of `'N'`, `'E'`, `'W'` or `'S'` (for North, East, West or South respectively), in this order. There will always be at least one possible move. If the current room is the exit room, the string ends with a single `'*'`.
2. Output: Your output, on a single line, must then be one of the following:
   (a) A character from `'N'`, `'E'`, `'W'`, `'S'`, indicating that you make one of the possible moves. After this, go back to 1 (in the new room).
   (b) An integer number which is your answer for this test case. This must be the minimum possible number of rooms you need to enter before reaching the exit room, or -1 if the exit room cannot be reached. This ends the sequence of input and output for this test case.

If your program outputs an invalid move or any other invalid output, you will receive a Wrong Answer.

Notes

C/C++ users:

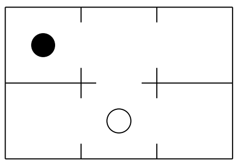
- Do `fflush(NULL);` after you print output.
- If you use `scanf` to read the input, then do not read newline characters. So, to read an integer n with `scanf`, use `scanf("%d", &n);`.

Java users:

- Do not use *buffered* output, `System.out.println` will do fine.

Example

Let us for example consider the following maze. The open circle represents the starting room, the filled circle represents the exit room.



Note that there might be multiple ways to find the correct answer, the sequence below is just an example. Extra newlines in the example below are for clarity only; you should print exactly one newline character after each line of output.

```
Input                                        Output

1
NEW
                                             N
EWS
                                             E
W
                                             W
EWS
                                             W
E*
                                             2
```

G Snooker

Problem

Ronnie owns a very old TV with no sound, and a screen so unclear that letters cannot be read. So when Ronnie watches a Snooker match on TV, he cannot see the current score, nor the currently active player. The only thing he can observe is the colour of the ball that is potted or missed. Ronnie does not care who wins, but he has no interest in decided matches. We therefore want to determine when a match is decided.

Snooker is a two-player game, played on a table with 15 red balls (each worth 1 point), and 1 yellow, green, brown, blue, pink and black ball (worth 2,3,4,5,6 and 7 points, respectively). Players take turns in which they score ("pot") a sequence of balls, and the turn changes when a player misses a ball. For both players, the initial goal in each turn is to first pot one red ball. When the player succeeds, he must pot a non-red ball of choice, after which he has to pot a red ball again, then again a non-red ball, etc. Red balls remain potted, and non-red balls are returned to the table when potted. When a player pots the last red ball, he must again (try to) pot a non-red ball, which then is returned to the table.

Next, the non-red balls are to be potted in their order of value (from 2 to 7), and are no longer returned to the table. The game has ended when the last ball (i.e., the black ball) has been potted, leaving an empty table. It is easily verified that it takes at least 36 shots to end the game.

The player with the highest score wins. If the scores are equal after the last ball has been potted, turn does not change and the black ball is re-placed on the table, and the first player to pot this ball wins. We assume that the only mistake the players can make is to miss a ball. In particular, we assume that they never pot the wrong ball (as can happen in the real game).

We call a game **decided**, if at some point during the game the difference between the scores of the two players is so big, that the player with the lower score cannot possibly win anymore.

Example 1: If the score is 60–44 and only the black and pink ball are left on the table, the difference is 16 and the value of the remaining balls is 13, and we call the game decided.

Example 2: If a player just scored a non-red ball, and there are two red balls and thus also all non-red balls remaining, the maximum value of the remaining balls is 1(red)+7(black)+1(red)+7 (black)+2+3+4+5+6+7= 43, so if the current score is 70–26, we call the game decided, but if the score is 70–28 or even 70–27, we do 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 $36 \leq N \leq 1,000$: the length of a sequence of balls that are hit.
- One line with N integers v satisfying $0 \leq v \leq 7$: the values of the balls that are potted successively, where $v = 0$ indicates that a player misses some ball.

  The integers constitute a complete, valid sequence. That is, they represent a game according to the rules described above, ending with the last, black ball. The integers are separated by single spaces.

Output

For every test case in the input, the output should contain a single number, on a single line: the (minimum) number i of the ball in the sequence (which is numbered from 1 to N), after which the game is decided.

Example

The first example below corresponds to the first example in the text, with a score 60–44 after ball 37.

Input

```
3
40
1 6 1 2 1 7 0 1 7 1 5 1 6 1 7 1 7 1 2 1 0 1 7 1 3 1 5 1 4 1 7 2 0 3 0 4 5 6 0 7
36
1 7 1 3 1 7 1 5 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 2 3 4 5 6 7
37
1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 0 1 0 1 7 1 7 1 7 1 7 1 7 2 3 4 5 6 7
```

Output

```
37
20
21
```

H Pie Division

Problem

Frans is celebrating his birthday. At an exclusive baker's shop, he has bought two delicious pies, of different types. He cuts each pie into a number of pieces. At coffee time, he invites his colleagues for a piece of pie, but after the celebration, some pieces are left over. In fact, $2N$ pieces are left over: N pieces of each pie, and N happens to be even. Frans does not want to take all remaining pieces back home, so he decides to share them with a colleague who is fond of pie too.

Now Frans wonders how to split the $2N$ remaining pieces, which seem to be randomly distributed over the table, in two. A simple way to achieve this would be to stretch a cord over the table, in a straight line. The pieces at one side of the cord are for Frans, the pieces at the other side are for the colleague. There is, however, a constraint. Both Frans and his colleague should take home $\frac{1}{2}N$ pieces of the first pie, and $\frac{1}{2}N$ pieces of the second pie. Is this possible with the cord trick? And if so, how many ways are there to do this? Of course, this depends on the positions of the $2N$ pieces on the table.

For example, let $N = 2$, and let the pieces of one pie be depicted by closed circles, and the pieces of the other pie by open circles. In the configuration of Figure 1(a), there are two possible divisions, as indicated in Figure 1(b) and Figure 1(c).

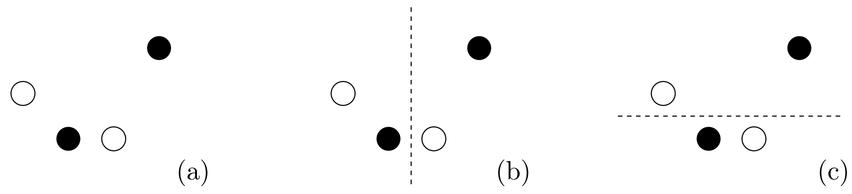


Figure 1: Two valid divisions of four pieces.

On the other hand, in the configuration of Figure 2(a), there is only one valid division, as indicated in Figure 2(b).

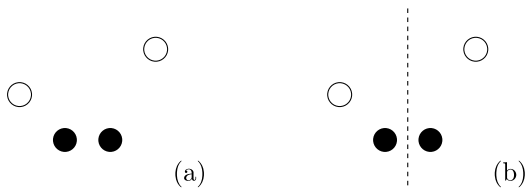


Figure 2: One valid division of four pieces.

To simplify things, we assume that no three pieces of pie are on the same line. This implies in particular that no two pieces of pie occupy the same position. Moreover, we assume that each piece is infinitely small.

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 even integer N, satisfying $2 \leq N \leq 1,000$.
- N lines, each with two integers x and y, satisfying $-10,000 \leq x, y \leq 10,000$: the x- and y-coordinates of a piece of the first pie.
- N lines, each with two integers x and y, satisfying $-10,000 \leq x, y \leq 10,000$: the x- and y-coordinates of a piece of the second pie.

Integers on the same line are separated by a single space.

Output

For every test case in the input, the output should contain a single number, on a single line: the number of ways to split the $2N$ pieces of pie in two, by stretching a cord over the table, such that at each side of the cord, there are $\frac{1}{2}N$ pieces of the first pie and $\frac{1}{2}N$ pieces of the second pie.

Example

The first two examples below correspond to the examples in the text.

Input

```
3
2
2 1
4 3
1 2
3 1
2
2 1
3 1
1 2
4 3
4
2 9
6 1
12 4
11 8
0 2
15 6
8 12
1 5
```

Output

```
2
1
3
```

I Keylogger

Problem

As a malicious hacker you are trying to steal your mother's password, and therefore you have installed a keylogger on her PC (or Mac, so you like). You have a log from your mother typing the password, but unfortunately the password is not directly visible because she used the left and right arrows to change the position of the cursor, and the backspace to delete some characters. Write a program that can decode the password from the given keylog.

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 L, satisfying $1 \leq \mathit{Length}(L) \leq 1,000,000$, consisting of:
  - `'-'` representing backspace: the character directly before the cursor position is deleted, if there is any.
  - `'<'` (and `'>'`) representing the left (right) arrow: the cursor is moved 1 character to the left (right), if possible.
  - alphanumeric characters, which are part of the password, unless deleted later. We assume 'insert mode': if the cursor is not at the end of the line, and you type an alphanumeric character, then all characters after the cursor move one position to the right.

Every decoded password will be of length > 0.

Output

For every test case in the input, the output should contain a single string, on a single line: the decoded password.

Example

Input

```
2
<<BP<A>>Cd-
ThIsIsS3Cr3t
```

Output

```
BAPC
ThIsIsS3Cr3t
```

This page intentionally left blank.

J Wrong Answer

Problem

You are solving a crossword puzzle and you have already written down the answers to all questions. While filling in the answers in the diagram, you notice that some answers have overlapping letters that do not match, so you must have made some mistakes in the answers. Instead of checking your answers, you decide to write a computer program to find the maximum number of answers that could have been correct.

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 H and V, satisfying $1 \leq H, V \leq 500$: the number of horizontal and vertical words, respectively.
- H lines, one for each horizontal word, each with two integers x and y and a string W, satisfying $0 \leq x, y \leq 1,000$ and $1 \leq \mathit{Length}(W) \leq 1,000$: the location of the first letter and the answer.
- V lines, one for each vertical word, each with two integers x and y and a string W, satisfying $0 \leq x, y \leq 1,000$ and $1 \leq \mathit{Length}(W) \leq 1,000$: the location of the first letter and the answer.

Integers and strings on the same line are separated by single spaces. No pair of horizontal words will overlap, nor will any pair of vertical words. The words consist of upper case letters only.

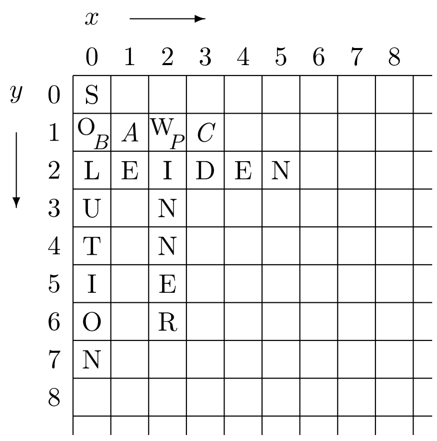
The top left corner of the diagram is at $x = y = 0$, x runs in the horizontal direction, and y runs in the vertical direction (downwards).

Output

For every test case in the input, the output should contain a single number, on a single line: the maximum number of answers that can fit in the crossword.

Example

The first example below corresponds to the following diagram, where the (assumed) wrong answer is written in italics:



Input

```
2
2 2
0 1 BAPC
0 2 LEIDEN
0 0 SOLUTION
2 1 WINNER
1 4
0 1 HELLO
1 0 HI
2 0 BYE
3 0 GOODBYE
4 0 FAREWELL
```

Output

```
3
4
```