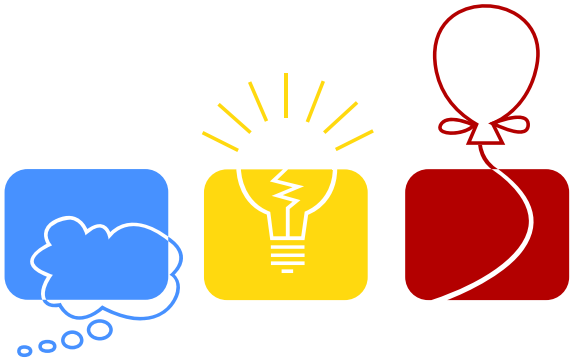
NWERC 2008

The 2008 ACM Northwestern Europe Programming Contest
Utrecht University, The Netherlands



acm International Collegiate Programming Contest

IBM event sponsor

The Problem Set

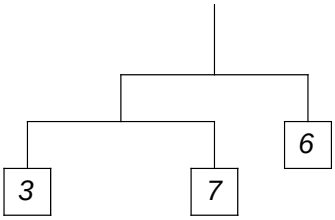
A	Equilibrium Mobile
B	Proving Equivalences
C	Cat vs. Dog
D	Disgruntled Judge
E	Easy Climb
F	Sculpture
G	***not available***
H	Matchsticks
I	White Water Rafting
J	Shuffle
K	Videopoker

Almost blank page

A Equilibrium Mobile

A mobile is a type of kinetic sculpture constructed to take advantage of the principle of equilibrium. It consists of a number of rods, from which weighted objects or further rods hang. The objects hanging from the rods balance each other, so that the rods remain more or less horizontal. Each rod hangs from only one string, which gives it freedom to rotate about the string.

We consider mobiles where each rod is attached to its string exactly in the middle, as in the figure underneath. You are given such a configuration, but the weights on the ends are chosen incorrectly, so that the mobile is not in equilibrium. Since that's not aesthetically pleasing, you decide to change some of the weights.



What is the minimum number of weights that you must change in order to bring the mobile to equilibrium? You may substitute any weight by any (possibly non-integer) weight. For the mobile shown in the figure, equilibrium can be reached by changing the middle weight from 7 to 3, so only 1 weight needs to changed.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with the structure of the mobile, which is a recursively defined expression of the form:

$$\langle \texttt{expr} \rangle \;::=\; \langle \texttt{weight} \rangle \;\mid\; \text{“[”}\, \langle \texttt{expr} \rangle\, \text{“,”}\, \langle \texttt{expr} \rangle\, \text{“]”}$$

  with $\langle \texttt{weight} \rangle$ a positive integer smaller than 10^9 indicating a weight and $[\langle \texttt{expr} \rangle, \langle \texttt{expr} \rangle]$ indicating a rod with the two expressions at the ends of the rod. The total number of rods in the chain from a weight to the top of the mobile will be at most 16.

Output

Per testcase:

- One line with the minimum number of weights that have to be changed.

Sample in- and output

Input	Output
3 [[3,7],6] 40 [[2,3],[4,5]]	1 0 3

B Proving Equivalences

Consider the following exercise, found in a generic linear algebra textbook.

> Let A be an $n \times n$ matrix. Prove that the following statements are equivalent:
>
> (a) A is invertible.
> (b) $Ax = b$ has exactly one solution for every $n \times 1$ matrix b.
> (c) $Ax = b$ is consistent for every $n \times 1$ matrix b.
> (d) $Ax = 0$ has only the trivial solution $x = 0$.

The typical way to solve such an exercise is to show a series of implications. For instance, one can proceed by showing that (a) implies (b), that (b) implies (c), that (c) implies (d), and finally that (d) implies (a). These four implications show that the four statements are equivalent.

Another way would be to show that (a) is equivalent to (b) (by proving that (a) implies (b) and that (b) implies (a)), that (b) is equivalent to (c), and that (c) is equivalent to (d). However, this way requires proving six implications, which is clearly a lot more work than just proving four implications!

I have been given some similar tasks, and have already started proving some implications. Now I wonder, how many more implications do I have to prove? Can you help me determine this?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line containing two integers n ($1 \leq n \leq 20\,000$) and m ($0 \leq m \leq 50\,000$): the number of statements and the number of implications that have already been proved.
- m lines with two integers s_1 and s_2 ($1 \leq s_1, s_2 \leq n$ and $s_1 \neq s_2$) each, indicating that it has been proved that statement s_1 implies statement s_2.

Output

Per testcase:

- One line with the minimum number of additional implications that need to be proved in order to prove that all statements are equivalent.

Sample in- and output

Input	Output
2 4 0 3 2 1 2 1 3	4 2

Almost blank page

C Cat vs. Dog

The latest reality show has hit the TV: "Cat vs. Dog". In this show, a bunch of cats and dogs compete for the very prestigious BEST PET EVER title. In each episode, the cats and dogs get to show themselves off, after which the viewers vote on which pets should stay and which should be forced to leave the show.

Each viewer gets to cast a vote on two things: one pet which should be kept on the show, and one pet which should be thrown out. Also, based on the universal fact that everyone is either a cat lover (i.e. a dog hater) or a dog lover (i.e. a cat hater), it has been decided that each vote must name exactly one cat and exactly one dog.

Ingenious as they are, the producers have decided to use an advancement procedure which guarantees that as many viewers as possible will continue watching the show: the pets that get to stay will be chosen so as to maximize the number of viewers who get both their opinions satisfied. Write a program to calculate this maximum number of viewers.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with three integers c, d, v ($1 \leq c, d \leq 100$ and $0 \leq v \leq 500$): the number of cats, dogs, and voters.
- v lines with two pet identifiers each. The first is the pet that this voter wants to keep, the second is the pet that this voter wants to throw out. A pet identifier starts with one of the characters '`C`' or '`D`', indicating whether the pet is a cat or dog, respectively. The remaining part of the identifier is an integer giving the number of the pet (between 1 and c for cats, and between 1 and d for dogs). So for instance, "`D42`" indicates dog number 42.

Output

Per testcase:

- One line with the maximum possible number of satisfied voters for the show.

Sample in- and output

Input	Output
2 1 1 2 C1 D1 D1 C1 1 2 4 C1 D1 C1 D1 C1 D2 D2 C1	1 3

Almost blank page

D Disgruntled Judge

Once upon a time, there was an NWERC judge with a tendency to create slightly too hard problems. As a result, his problems were never solved. As you can image, this made our judge somewhat frustrated. This year, this frustration has culminated, and he has decided that rather than spending a lot of time constructing a well-crafted problem, he will simply write some insanely hard problem statement and just generate some random input and output files. After all, why bother having proper test data if nobody is going to try the problem anyway?

Thus, the judge generates a testcase by simply letting the input be a random number, and letting the output be another random number. Formally, to generate the data set with T test cases, the judge generates $2T$ random numbers $x_1, \ldots, x_{2T}$ between 0 and $10\,000$, and then writes T, followed by the sequence $x_1, x_3, x_5, \ldots, x_{2T-1}$ to the input file, and the sequence $x_2, x_4, x_6, \ldots, x_{2T}$ to the output file.

The random number generator the judge uses is quite simple. He picks three numbers x_1, a, and b between 0 and $10\,000$ (inclusive), and then for i from 2 to $2T$ lets

$$x_i = (a \cdot x_{i-1} + b) \bmod 10\,001.$$

You may have thought that such a poorly designed problem would not be used in a contest of such high standards as NWERC. Well, you were wrong.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line containing an integer n ($0 \leq n \leq 10\,000$): an input testcase.

The input file is guaranteed to be generated by the process described above.

Output

Per testcase:

- One line with an integer giving the answer for the testcase.

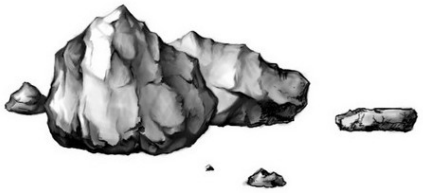
If there is more than one output file consistent with the input file, any one of these is acceptable.

Sample in- and output

Input	Output
3 17 822 3014	9727 1918 4110

Almost blank page

E Easy Climb



Somewhere in the neighborhood we have a very nice mountain that gives a splendid view over the surrounding area. There is one problem though: climbing this mountain is very difficult, because of rather large height differences. To make more people able to climb the mountain and enjoy the view, we would like to make the climb easier.

To do so, we will model the mountain as follows: the mountain consists of n adjacent stacks of stones, and each of the stacks is h_i high. The successive height differences are therefore $h_{i+1} - h_i$ (for $1 \leq i \leq n-1$). We would like all absolute values of these height differences to be smaller than or equal to some number d.

We can do this by increasing or decreasing the height of some of the stacks. The first stack (the starting point) and the last stack (the ending point) should remain at the same height as they are initially. Since adding and removing stones requires a lot of effort, we would like to minimize the total number of added stones plus the total number of removed stones. What is this minimum number?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with two integers n ($2 \leq n \leq 100$) and d ($0 \leq d \leq 10^9$): the number of stacks of stones and the maximum allowed height difference.
- One line with n integers h_i ($0 \leq h_i \leq 10^9$): the heights of the stacks.

Output

Per testcase:

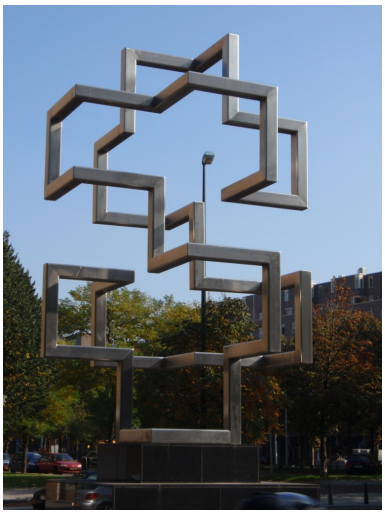
- One line with the minimum number of stones that have to be added or removed or "`impossible`" if it is impossible to achieve the goal.

Sample in- and output

Input	Output
3 10 2 4 5 10 6 6 9 4 7 9 8 3 1 6 4 0 4 2 3 0 6 3	6 impossible 4

Almost blank page

F Sculpture



Imagine a box, made of copper plate. Imagine a second one, intersecting the first one, and several others, intersecting each other (or not). That is how the sculptor Oto Boxing constructs his sculptures. In fact he does not construct that much, he only makes the design; the actual construction is contracted out to a construction company. For the calculation of the costs of construction the company needs to know the total area of copper plate involved. Parts of a box that are hidden in another box are not realized in copper, of course. (Copper is quite expensive, and prices are rising.) After construction, the total construction is plunged into a bath of chemicals. To prevent this bath from running over, the construction company wants to know the total volume of the construction. Given that a construction is a collection of boxes, you are asked to calculate the area and the volume of the construction.

Some of Oto's designs are connected, others are not. Either way, we want to know the total area and the total volume. It might happen that the boxes completely enclose space that is not included in any of the boxes (see the second example below). Because the liquid cannot enter that space, its volume must be added to the total volume. Copper plate bordering this space is superfluous, of course, so it does not add to the area.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with an integer n ($1 \leq n \leq 50$): the number of boxes involved.
- n lines with six positive integers x_0, y_0, z_0, x, y, z ($1 \leq x_0, y_0, z_0, x, y, z \leq 500$): the triple (x_0, y_0, z_0) is the vertex of the box with the minimum values for the coordinates and the numbers x, y, z are the dimensions of the box (x, y and z dimension, respectively). All dimensions are in centimeters. The sides of the boxes are always parallel to the coordinate axes.

Output

Per testcase:

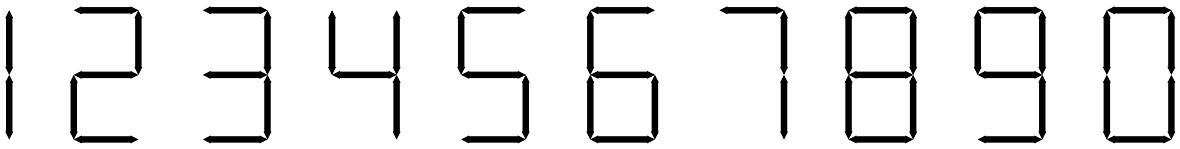
- One line with two numbers separated by single spaces: the total amount of copper plate needed (in cm^2), and the total volume (in cm^3).

Sample in- and output

Input	Output
2 2 1 2 3 3 4 5 6 2 3 3 4 5 7 1 1 1 5 5 1 1 1 10 5 5 1 1 1 2 1 4 8 2 1 2 4 1 8 5 2 2 1 4 8 1 5 2 4 1 8 3 3 4 1 1 1	188 120 250 250

H Matchsticks

Matchsticks are ideal tools to represent numbers. A common way to represent the ten decimal digits with matchsticks is the following:



This is identical to how numbers are displayed on an ordinary alarm clock. With a given number of matchsticks you can generate a wide range of numbers. We are wondering what the smallest and largest numbers are that can be created by using all your matchsticks.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with an integer n ($2 \leq n \leq 100$): the number of matchsticks you have.

Output

Per testcase:

- One line with the smallest and largest numbers you can create, separated by a single space. Both numbers should be positive and contain no leading zeroes.

Sample in- and output

Input	Output
4 3 6 7 15	7 7 6 111 8 711 108 7111111

Almost blank page

I White Water Rafting

You have been hired by a big theme park to design a new attraction: a white water rafting ride. You already designed the track; it is a round trip that is described by an inner and an outer polygon. The space in between the two polygons is the track.

You still need to design the rafts, however. It has been decided that they should be circular, so that they can spin freely along the track and increase the fun and excitement of the ride. Besides that, they should be as big as possible to fit the maximum number of people, but they can't be too big, for otherwise they would get stuck somewhere on the track.

What is the maximum radius of the rafts so that they can complete the track?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with an integer n_i ($3 \leq n_i \leq 100$): the number of points of the inner polygon.
- n_i lines with two integers each: the coordinates of the points of the inner polygon in consecutive order.
- One line with an integer n_o ($3 \leq n_o \leq 100$): the number of points of the outer polygon.
- n_o lines with two integers each: the coordinates of the points of the outer polygon in consecutive order.

All coordinates have absolute value no larger than $1\,000$. The points of the polygons can be given in either clockwise or counterclockwise order and the two polygons do not intersect or touch themselves or each other. The outer polygon encloses the inner polygon.

Output

Per testcase:

- One line with a floating point number: the maximal radius of the white water rafts. This number must have a relative or absolute error less than 10^{-6}.

Sample in- and output

Input	Output
2 4 -5 -5 5 -5 5 5 -5 5 4 -10 -10 -10 10 10 10 10 -10 3 0 0 1 0 1 1 5 3 -3 3 3 -4 2 -1 -1 -2 -2	2.5 0.70710678

J Shuffle

You are listening to your music collection using the shuffle function to keep the music surprising. You assume that the shuffle algorithm of your music player makes a random permutation of the songs in the playlist and plays the songs in that order until all songs have been played. Then it reshuffles and starts playing the list again.

You have a history of the songs that have been played. However, your record of the history of played songs is not complete, as you started recording songs at a certain point in time and a number of songs might already have been played. From this history, you want to know at how many different points in the future the next reshuffle might occur.

A potential future reshuffle position is valid if it divides the recorded history into intervals of length s (the number of songs in the playlist) with the first and last interval possibly containing less than s songs and no interval contains a specific song more than once.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with two integers s and n ($1 \leq s, n \leq 100\,000$): the number of different songs in the playlist and the number of songs in the recorded playlist history.
- One line with n space separated integers, $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq s$): the recorded playlist history.

Output

Per testcase:

- One line with the number of future positions the next reshuffle can be at. If the history could not be generated by the above mentioned algorithm, output 0.

Sample in- and output

Input	Output
4 4 10 3 4 4 1 3 2 1 2 3 4 6 6 6 5 4 3 2 1 3 5 3 3 1 1 1 7 3 5 7 3	1 6 0 7

Almost blank page

K Videopoker

Videopoker is the slot machine variant of the currently immensely popular game of poker. It is a variant on draw poker. In this game the player gets a hand consisting of five cards randomly drawn from a standard 52-card deck. From this hand, the player may discard any number of cards (between 0 and 5, inclusive), and change them for new cards randomly drawn from the remainder of the deck. After that, the hand is evaluated and the player is rewarded according to a payout structure. A common payout structure is as follows:

hand	payout
one pair	1
two pair	2
three of a kind	3
straight	4
flush	5
full house	10
four of a kind	25
straight flush	100
royal flush	250

Once you know the payout structure, you can determine for a given hand which cards you must change to maximize your expected reward. We'd like to know this expected reward, given a hand.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with nine integers x_i ($0 \leq x_i \leq 1\,000$) describing the payout structure. The numbers are in increasing order and describe the payout for one pair, two pair, etc, until the royal flush.
- One line with one integer n ($1 \leq n \leq 10$): the number of starting hands to follow.
- n lines, each describing a starting hand. A hand consists of five space separated tokens of the form `Xs`, with `X` being the rank ('`2`' ... '`9`', '`T`', '`J`', '`Q`', '`K`' or '`A`') and `s` being the suit ('`c`', '`d`', '`h`' or '`s`').

Output

Per testcase:

- One line for each starting hand with a floating point number that is the maximal expected reward for that hand. These numbers must have an absolute or relative error less than 10^{-6}.

Sample in- and output

Input	Output
1 1 2 3 4 5 10 25 100 250 5 Ah Ac Ad As 2s Ks Qs Js Ts 2h Ks Qs 2d 2h 3s 2d 4h 5d 3c 9c 2h 3h 6d 8h Tc	25.000000 8.9574468 1.5467160 0.9361702 0.6608135

Poker hand rankings

For those of you not familiar with the game of poker, here follow explanations of the different poker hand rankings:

- "one pair"consists of two cards of the same rank and three unmatched cards, e.g. Ah As Tc 8h 2c;
- "two pair" consists of two cards of the same rank, two cards of a different same rank and an unmatched card, e.g. Ah As Th Ts 3c;
- "three of a kind" consists of three cards of the same rank and two unmatched cards, e.g. Kc Kh Ks 6c 5s;
- a "straight" consists of five cards of sequential rank in more than one suit, e.g. Jd Ts 9c 8d 7h. The ace can be used as low or high card, so straights from ace to five and from ten to ace can be formed;
- a "flush" consists of five cards of the same suit, that are not in sequential rank, e.g. Ks Qs 8s 5s 3s;
- a "full house" consists of three cards of the same rank and two cards of a different same rank, e.g. Js Jh Jc 4s 4c;
- "four of a kind" consists of four cards of the same rank and an unmatched card, e.g. 7h 7c 7s 7d 5c;
- a "straight flush" consists of five cards that form both a straight and a flush and that is not a royal flush, e.g. 7h 6h 5h 4h 3h;
- a "royal flush" consists of five cards that form both a straight from ten to ace and a flush, e.g. As Ks Qs Js Ts.