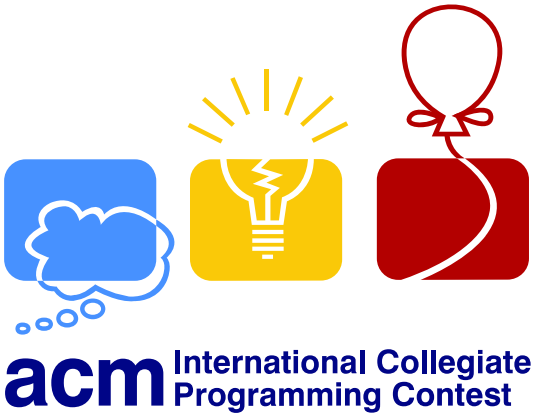
The 2009 ACM North Western European Regional Contest
Friedrich-Alexander-University, Nuremberg, Germany

November 8, 2009



The Problem Set

Good luck and have fun!

This page is intentionally left (almost) blank.

Problem A
An Industrial Spy

Industrial spying is very common for modern research labs. I am such an industrial spy – don't tell anybody! My recent job was to steal the latest inventions from a famous math research lab. It was hard to obtain some of their results but I got their waste out of a document shredder.
I have already reconstructed that their research topic is fast factorization. But the remaining paper snippets only have single digits on it and I cannot imagine what they are for. Could it be that those digits form prime numbers? Please help me to find out how many prime numbers can be formed using the given digits.

Input

The first line of the input holds the number of test cases c ($1 \leq c \leq 200$). Each test case consists of a single line. This line contains the digits (at least one, at most seven) that are on the paper snippets.

Output

For each test case, print one line containing the number of different primes that can be reconstructed by shuffling the digits. You may ignore digits while reconstructing the primes (e.g., if you get the digits 7 and 1, you can reconstruct three primes 7, 17, and 71). Reconstructed numbers that (regarded as strings) differ just by leading zeros, are considered identical (see the fourth case of the sample input).

Sample Input	Sample Output
4	3
17	1336
1276543	0
9999999	2
011	

This page is intentionally left (almost) blank.

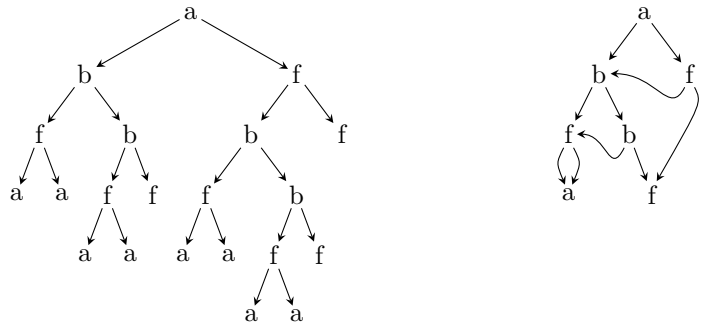
Problem B

Common Subexpression Elimination

Let the set Σ consist of all words composed of 1–4 lower case letters, such as the words "a", "b", "f", "aa", "fun" and "kvqf". Consider expressions according to the grammar with the two rules

$$E \to f$$
$$E \to f(E, E)$$

for every symbol $f \in \Sigma$. Any expression can easily be represented as a tree according to its syntax. For example, the expression "a(b(f(a,a),b(f(a,a),f)),f(b(f(a,a),b(f(a,a),f)),f))" is represented by the tree on the left in the following figure:



Last night you dreamt of a great invention which considerably reduces the size of the representation: use a graph instead of a tree, to share common subexpressions. For example, the expression above can be represented by the graph on the right in the figure. While the tree contains 21 nodes, the graph just contains 7 nodes.
Since the tree on the left in the figure is also a graph, the representation using graphs is not necessarily unique. Given an expression, find a graph representing the expression with as few nodes as possible!

Input

The first line of the input contains the number c ($1 \le c \le 200$), the number of expressions. Each of the following c lines contains an expression according to the given syntax, without any whitespace. Its tree representation contains at most 50 000 nodes.

Output

For each expression, print a single line containing a graph representation with as few nodes as possible.
The graph representation is written down as a string by replacing the appropriate subexpressions with numbers. Each number points to the root node of the subexpression which should be inserted at that position. Nodes are numbered sequentially, starting with 1; this numbering includes just the nodes of the graph (not those which have been replaced by numbers). Numbers must point to nodes written down before (no forward pointers). For our example, we obtain "a(b(f(a,4),b(3,f)),f(2,6))".

Sample Input

```
3
this(is(a,tiny),tree)
a(b(f(a,a),b(f(a,a),f)),f(b(f(a,a),b(f(a,a),f)),f))
z(zz(zzzz(zz,z),zzzz(zz,z)),zzzz(zz(zzzz(zz,z),zzzz(zz,z)),z))
```

Sample Output

```
this(is(a,tiny),tree)
a(b(f(a,4),b(3,f)),f(2,6))
z(zz(zzzz(zz,z),3),zzzz(2,5))
```

This page is intentionally left (almost) blank.

Problem C

Divisible Subsequences

Given a sequence of positive integers, count all contiguous subsequences (sometimes called *substrings*, in contrast to *subsequences*, which may leave out elements) the sum of which is divisible by a given number. These subsequences may overlap. For example, the sequence (see sample input)

$$2, 1, 2, 1, 1, 2, 1, 2$$

contains six contiguous subsequences the sum of which is divisible by four: the first to eighth number, the second to fourth number, the second to seventh number, the third to fifth number, the fourth to sixth number, and the fifth to seventh number.

Input

The first line of the input consists of an integer c ($1 \leq c \leq 200$), the number of test cases. Then follow two lines per test case.
Each test case starts with a line consisting of two integers d ($1 \leq d \leq 1\,000\,000$) and n ($1 \leq n \leq 50\,000$), the divisor of the sum of the subsequences and the length of the sequence, respectively. The second line of a test case contains the elements of the sequence, which are integers between 1 and $1\,000\,000\,000$, inclusively.

Output

For each test case, print a single line consisting of a single integer, the number of contiguous subsequences the sum of which is divisible by d.

Sample Input

```
2
7 3
1 2 3
4 8
2 1 2 1 1 2 1 2
```

Sample Output

```
0
6
```

This page is intentionally left (almost) blank.

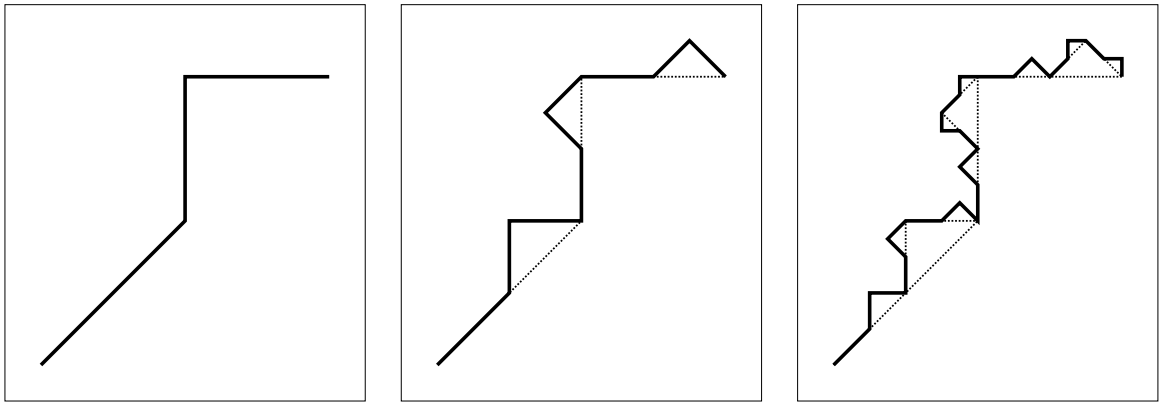
Problem D

Fractal

Fractals are really cool mathematical objects. They have a lot of interesting properties, often including:

- fine structure at arbitrarily small scales;
- self-similarity, i.e., magnified it looks like a copy of itself;
- a simple, recursive definition.

Approximate fractals are found a lot in nature, for example, in structures such as clouds, snow flakes, mountain ranges, and river networks.



In this problem, we consider fractals generated by the following algorithm: we start with a polyline, i.e., a set of connected line segments. This is what we call a fractal of depth one (see leftmost picture). To obtain a fractal of depth two, we replace each line segment with a scaled and rotated version of the original polyline (see middle picture). By repetitively replacing the line segments with the polyline, we obtain fractals of arbitrary depth and very fine structures arise. The rightmost picture shows a fractal of depth three.
The complexity of an approximate fractal increases quickly as its depth increases. We want to know where we end up after traversing a certain fraction of its length.

Input

The input starts with a single number c ($1 \leq c \leq 200$) on one line, the number of test cases. Then each test case starts with one line with n ($3 \leq n \leq 100$), the number of points of the polyline. Then follow n lines with on the ith line two integers x_i and y_i ($-1\,000 \leq x_i, y_i \leq 1\,000$), the consecutive points of the polyline. Next follows one line with an integer d ($1 \leq d \leq 10$), the depth of the fractal. Finally, there is one line with a floating point number f ($0 \leq f \leq 1$), the fraction of the length that is traversed.
The length of each line segment of the polyline is smaller than the distance between the first point (x_1, y_1) and the last point (x_n, y_n) of the polyline. The length of the complete polyline is smaller than twice this distance.

Output

Per test case, the output contains one line with the coordinate where we end up. Format it as `(x,y)`, with two floating point numbers x and y. The absolute error in both coordinates should be smaller than 10^{-6}.

Sample Input

```
1
4
-2 -2
0 0
0 2
2 2
3
0.75
```

Sample Output

```
(0.4267766953,2)
```

This page is intentionally left (almost) blank.

Problem E
Mountain Road

In the Franconian Switzerland, there is a narrow mountain road. With only a single lane, this is a bottleneck for two-way traffic. Your job is to schedule incoming cars at both ends so that the last car leaves the road as early as possible.

Each car is specified by three values: the direction in which it is going, the arrival time at the corresponding beginning of the road, and the driving time this car needs to get through, provided it is not slowed down by other cars in front. Cars cannot overtake each other on the mountain road, and reordering cars in the queues at the ends of the road is not allowed.

For safety reasons, two successive cars going in the same direction may not pass any point of the road within less than 10 seconds. This ensures that the second car will not crash into the first car if the latter brakes hard. However, if another car passes in the other direction in between, it will be clear that the road is empty, so in this case, this rule does not apply.

Input

The first line of the input consists of a single integer c ($1 \leq c \leq 200$), the number of test cases.

Then follow the test cases, each beginning with a single line consisting of an integer n ($1 \leq n \leq 200$), the number of cars you are to consider in this test case. The remainder of each test case consists of n lines, one line per car, starting with a single upper case letter ("`A`" or "`B`"), giving the direction in which the car is going. Then follow, on the same line, two integers t ($0 \leq t \leq 100\,000$) and d ($1 \leq d \leq 100\,000$), giving the arrival time at the beginning of the road and the minimum travel time, respectively, both in seconds.

Within a test case, the cars are given in order of increasing arrival time, and no two cars will arrive at the same time.

Output

For each test case, print a single line consisting of the point in time (in seconds) the last car leaves the road when the cars are scheduled optimally.

Sample Input	Sample Output
2 4 A 0 60 B 19 10 B 80 20 A 85 100 4 A 0 100 B 50 100 A 100 1 A 170 100	200 270

This page is intentionally left (almost) blank.

Problem F

Moving to Nuremberg

One of the most important inventions for modern-day city life is the public transportation. However, most people probably do not think of it that way – even though it makes travel in the city a lot easier, we generally want to spend as little time as possible on the subway.

After your experience in NWERC 2009, Nuremberg holds a special place in your heart, and some years later you decide to move here. Your only problem is to figure out which part of Nuremberg to move to. Naturally, you want to move to a nice neighborhood, but since most parts of the city are nice there are still a lot of choices. Being naturally averse to spending hours each day on commuting, you instead decide to choose a place based on the amount of time you will have to spend on the subway.

Now, if you were only going to go to one place, it would be easy to find the best place to live. But of course, there are several places where you anticipate that you will go regularly, such as work, friends, and the occasional Christkindlesmarkt. In order to be able to work this out, you have written a list of all the places which you want to visit regularly, along with estimates of how often you want to go there. For simplicity, you assume that you will always go somewhere and then back home, e.g., if you are going to Christkindlesmarkt after work you will drop by your house on the way from work before going to the markt, rather than going to the markt directly from work. Now, you have to find the place to live for which the total travel time is minimal.

Because Nuremberg has an extensive public transportation system, you will be using the subway for traveling. The subway net is quite big, but is still fairly easily maneuvered because it is shaped like a tree. In other words, there is always a unique path between any pair of subway stations. (This is not quite true for the Nuremberg subway of today, but when you move here in a few years, we anticipate that it will be true.)

Input

The input consists of several test cases. The first line of input contains an integer c ($1 \leq c \leq 200$), giving the number of test cases. Then, each test case starts with an integer n ($1 \leq n \leq 50\,000$), giving the number of subway stations in Nuremberg. Then follow $n-1$ lines, describing the subway net. Each of these lines contains three integers a, b, and t ($1 \leq a, b \leq n$, and $1 \leq t \leq 300$), indicating that stations a and b are adjacent and that it takes t seconds to travel between them. This is followed by a line containing an integer m ($0 \leq m \leq n$), denoting the number of stations which you want to go to regularly. Then follow m lines. Each of these lines contains two integers a and f ($1 \leq a \leq n$, $1 \leq f \leq 500$), where a is the station you want to visit and f is the number of times you want to visit this station in a year. No station will occur in this list more than once.

Output

For each test case, first output a line containing the number of seconds spent in traffic during a year, provided you choose an optimal place to live. Following this line, output a line giving all optimal choices of subway stations, separated by single spaces and in increasing order.

Sample Input

```
2
2
1 2 17
2
1 5
2 10
5
1 3 10
2 3 20
3 4 30
4 5 30
3
1 10
2 10
5 20
```

Sample Output

```
170
2
3000
3 4 5
```

This page is intentionally left (almost) blank.

Problem G
Room Assignments

Once there was an inventor congress, where inventors from all over the world met in one place. The organizer of the congress reserved exactly one hotel room for each inventor. Each inventor, however, had its own preference regarding which room he would like to stay in. Being a clever inventor himself, the organizer soon found an objective way of doing the room assignments in a fair manner: each inventor wrote two different room numbers on a fair coin, one room number on each side. Then, each inventor threw his coin and was assigned the room number which was shown on the upper side of his coin. If some room had been assigned to more than one inventor, all inventors had to throw their coins again.

As you can imagine, this assignment process could take a long time or even not terminate at all. It has the advantage, however, that among all possible room assignments, one assignment is chosen randomly according to a uniform distribution. In order to apply this method in modern days, you should write a program which helps the organizer.

The organizer himself needs a hotel room too. As the organizer, he wants to have some advantage: he should be able to rate each of the rooms (the higher the rating, the better), and the program should tell him which two room numbers he should write on his coin in order to maximize the expected rating of the room he will be assigned to. The program also has access to the choices of the other inventors before making the proposal. It should never propose two rooms for the organizer such that it is not possible to assign all inventors to the rooms, if a valid assignment is possible at all.

Input

The input starts with a single number c ($1 \leq c \leq 200$) on one line, the number of test cases. Each test case starts with one line containing a number n ($2 \leq n \leq 50\,000$), the number of inventors and rooms. The following $n-1$ lines contain the choices of the $n-1$ guests (excluding the organizer). For each inventor, there is a line containing two numbers a and b ($1 \leq a < b \leq n$), the two room numbers which are selected by the inventor. The last line of each test case consists of n integers $v_1, \ldots, v_n$ ($1 \leq v_i \leq 1\,000\,000$), where v_i is the organizer's rating for room i.

Output

For each test case, print a single line containing the two different room numbers a and b which should be selected by the organizer in order to maximize the expected rating of the room he will be assigned to. If there is more than one optimal selection, break ties by choosing the smallest a and, for equal a, the smallest b. If there is no way for the organizer to select two rooms such that an assignment of inventors to rooms is possible, print "`impossible`" instead.

Sample Input

```
3
4
1 2
2 3
1 3
2 3 4 1
3
1 2
2 3
100 40 70
5
1 2
1 2
1 2
3 4
1 1 1 1 1
```

Sample Output

```
1 4
1 3
impossible
```

This page is intentionally left (almost) blank.

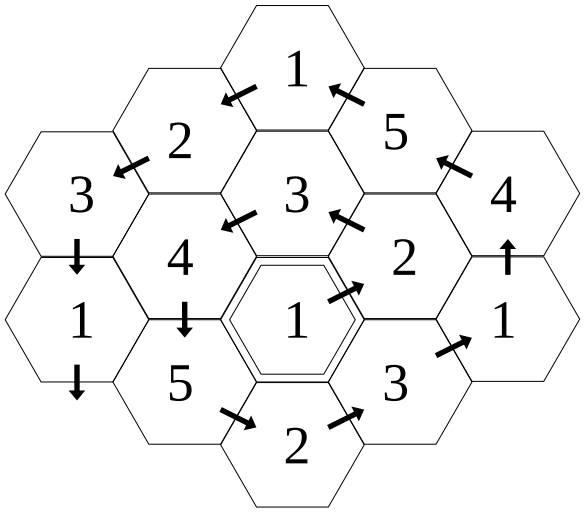
Problem H

Settlers of Catan

The popular board game "Settlers of Catan" starts by creating a random board. This board consists of hexagonal resource tiles containing five different resources: clay, lumber, wool, grain, and ore. For simplicity, we will denote these by the numbers 1 to 5.
Random boards, however, often have multiple equal resource tiles next to each other. This annoys some players. Therefore, we have invented a new way of creating the playing board. Starting in the middle and spiraling outwards, each time we add a new tile to the board we choose the resource of the tile according to the following rules:

- the new tile must be different from its neighboring tiles on the board so far;
- in case multiple tiles are possible, we choose a resource that occurs the least number of times on the board so far;
- in case multiple tiles are still possible, the new resource must have the lowest number possible.

The figure underneath shows how to spiral outwards and which resource tiles are chosen first. We are curious what the number of the resource is on the nth tile that is added to the board (starting with $n = 1$).



Input

On the first line of the input there is one integer c ($1 \leq c \leq 200$), the number of test cases. Each following test case consists of a single line with one integer n ($1 \leq n \leq 10\,000$), the number of the tile we are curious about.

Output

For each test case, print a single line with one integer, specifying the resource of the nth tile.

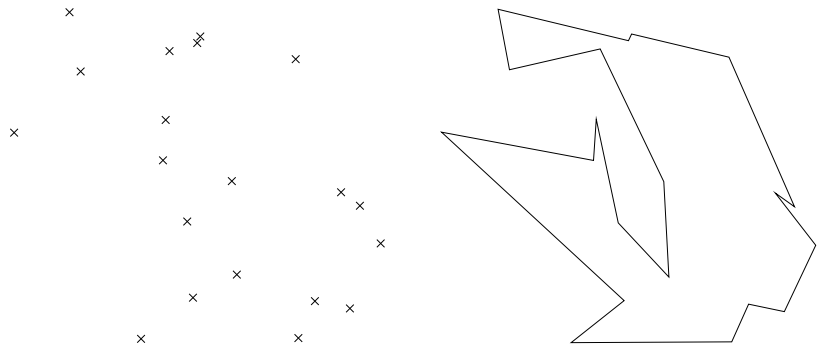
Sample Input	Sample Output
4	1
1	4
4	5
10	5
100	

This page is intentionally left (almost) blank.

Problem I

Simple Polygon

Write a program that constructs a polygon from a given set of points in the plane. Each point of the set has to be a vertex of the polygon, and the polygon must not have any other vertices. No two line segments of the polygon may have any point in common, except for the middle vertex of two consecutive line segments. For example, given the points on the left-hand side, a valid polygon is shown on the right-hand side:



A valid polygon is guaranteed to exist, and any valid polygon will be accepted as a correct answer. Moreover, no two points will share the same location, and not all points will be collinear.

Input

The first line of the input consists of an integer c ($1 \leq c \leq 200$), the number of test cases. Then follow the test cases, one per line.
Each test case starts with an integer n ($3 \leq n \leq 2\,000$), the number of given points. Then follow, on the same line, the coordinates of the points. Each point is specified by two integers x and y in the range $-10\,000$ to $10\,000$, inclusively.

Output

For each test case, print a single line containing a permutation of the numbers 0 to $n - 1$. Each of these numbers represents the index of a point, in the same order as given in the input. When drawing line segments between consecutive points in the order given by this permutation, the result must be a valid polygon.

Sample Input

```
2
4 0 0 2 0 0 1 1 0
5 0 0 10 0 10 5 5 -1 0 5
```

Sample Output

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

This page is intentionally left (almost) blank.

Problem J
Wormholes

A friend of yours, an inventor, has built a spaceship recently and wants to explore space with it. During his first voyages, he discovered that the universe is full of wormholes created by some alien race. These wormholes allow one to travel to places far, far away, but moreover, they can also send you to times long ago or in the distant future.

Having mapped these wormholes and their respective end points, you and your friend boldly decide to board his spaceship and go to some distant place you'd like to visit. Of course, you want to arrive at your destination as early as possible. The question is: what is this earliest arrival time?

Input

The first line of input contains an integer c ($1 \leq c \leq 200$), the number of test cases. Each test case starts with a line containing two coordinate triples x_0, y_0, z_0 and x_1, y_1, z_1, the space coordinates of your departure point and destination. The next line contains an integer n ($0 \leq n \leq 50$), the number of wormholes. Then follow n lines, one for each wormhole, with two coordinate triples x_s, y_s, z_s and x_e, y_e, z_e, the space coordinates of the wormhole entry and exit points, respectively, followed by two integers t, d ($-1\,000\,000 \leq t, d \leq 1\,000\,000$), the creation time t of the wormhole and the time shift d when traveling through the wormhole.

All coordinates are integers with absolute values smaller than or equal to 10 000 and no two points are the same.

Note that, initially, the time is zero, and that tunneling through a wormhole happens instantly. For simplicity, the distance between two points is defined as their Euclidean distance (the square root of the sum of the squares of coordinate differences) rounded up to the nearest integer. Your friend's spaceship travels at speed 1.

Output

For each test case, print a single line containing an integer: the earliest time you can arrive at your destination.

Sample Input

```
2
0 0 0 100 0 0
2
1 1 0 1 2 0 -100 -2
0 1 0 100 1 0 -150 10
0 0 0 10 0 0
1
5 0 0 -5 0 0 0 0
```

Sample Output

```
-89
10
```

This page is intentionally left (almost) blank.