Colliding Traffic

For a boat on a small, constrained body of water, other traffic can be a major hazard. The more traffic there is in the same area, the higher the risk of a collision.

Your job is to monitor traffic and help detect likely collisions before they occur. You have sensors to detect the position, direction, and speed of each boat. Assuming the direction and speed remain constant, your task is to determine whether any of the boats will collide. Two boats are considered to collide if they come within a given distance of each other.

Input Specification

You have to read the input from a text file "CollidingTraffic.txt". The first line of input contains a single integer *c*, the number of test cases to follow. Each test case starts with a line containing two numbers, *n*, the number of boats, and *r*, the collision distance. Two boats are considered to collide if they come within *r* meters of each other. There will be no more than 1000 boats. Each boat is identified by a line containing four numbers *x*, *y*, *d*, *s*. The numbers *x* and *y* give the current position of the boat as a distance east and north, respectively, from a common origin, and will be between -1000 and 1000, inclusive. The lake is small enough that we can model it as a flat surface. The number *d* gives the direction in which the boat is heading in degrees clockwise from north (so east is 90 degrees). The number *s* gives the speed of the boat in metres per second, and will be between 0.001 and 1000. Note that *r*, *x*, *y*, *d*, and *s* are not necessarily integers. The input data will be such that the answer will not change if any of the numbers *x*,*y*, *d* and *s* are changed by 10^-6 or less.

Sample Input

```
2
2 5
0 0 90 1
10 10 180 1
2 10
0 0 0 0
8 8 270 1
```

Output Specification

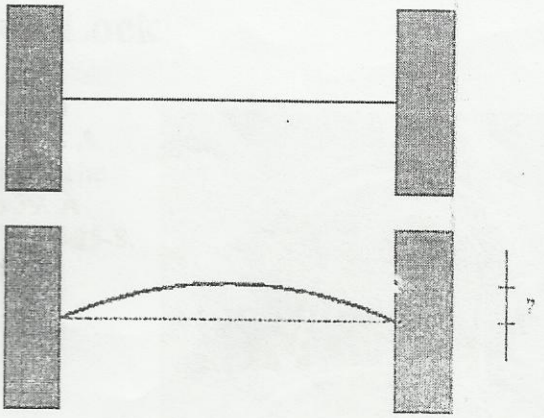
For each test case, output a line containing a single integer, the number of seconds, rounded to the nearest second, before any of the boats come within *r* metres of each other. If none of the boats ever collide, output the line:

```
No collision.
```

Output for Sample Input

```
6
2
```

Expanding Rods



When a thin rod of length L is heated n degrees, it expands to a new length $L'=(1+n*C)*L$, where C is the coefficient of heat expansion.

When a thin rod is mounted on two solid walls and then heated, it expands and takes the shape of a circular segment, the original rod being the chord of the segment.

Your task is to compute the distance by which the center of the rod is displaced.

You have to read the input from a text file "ExpandingRods.txt". The input contains multiple lines. Each line of input contains three non-negative numbers: the initial lenth of the rod in millimeters, the temperature change in degrees and the coefficient of heat expansion of the material. Input data guarantee that no rod expands by more than one half of its original length. The last line of input contains three negative numbers and it should not be processed.

For each line of input, output one line with the displacement of the center of the rod in millimeters with 3 digits of precision.

Sample input

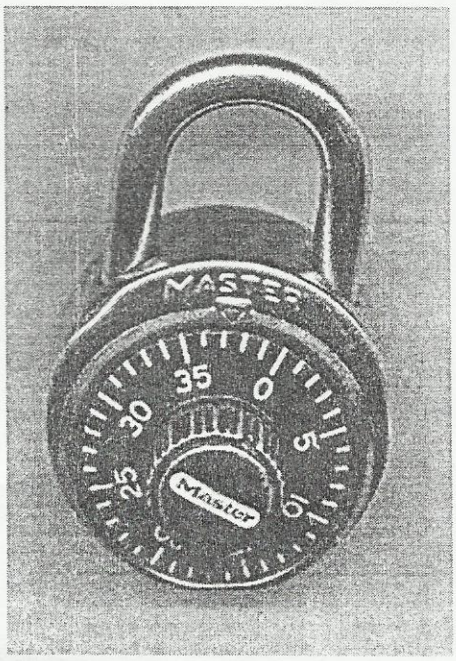
1000 100 0.0001
15000 10 0.00006
10 0 0.001
-1 -1 -1

Output for sample input

61.329
225.020
0.000

Combination Lock

Now that you're back to school for another term, you need to remember how to work the combination lock on your locker. A common design is that of the Master Brand, shown at right. The lock has a dial with 40 calibration marks numbered 0 to 39. A combination consists of 3 of these numbers; for example: 15-25-8. To open the lock, the following steps are taken:



- turn the dial clockwise 2 full turns
- stop at the first number of the combination
- turn the dial counter-clockwise 1 full turn
- continue turning counter-clockwise until the 2nd number is reached
- turn the dial clockwise again until the 3rd number is reached
- pull the shank and the lock will open.

Given the initial position of the dial and the combination for the lock, how many degrees is the dial rotated in total (clockwise plus counter-clockwise) in opening the lock?

You have to read the input from a text file "CombinationLock.txt". Input consists of several test cases. For each case there is a line of input containing 4 numbers between 0 and 39. The first number is the position of the dial. The next three numbers are the combination. Consecutive numbers in the combination will be distinct. A line containing 0 0 0 0 follows the last case.

For each case, print a line with a single integer: the number of degrees that the dial must be turned to open the lock.

Sample Input

```
0 30 0 30
5 35 5 35
0 20 0 20
7 27 7 27
0 10 0 10
9 19 9 19
0 0 0 0
```

Output for Sample Input

```
1350
1350
1620
1620
1890
1890
```

ProBrains-2012@UOK

Speed Programming

Round - 1

Exact Change

- Seller: That will be fourteen dollars.
- Buyer: Here's a twenty.
- Seller: Sorry, I don't have any change.
- Buyer: OK, here's a ten and a five. Keep the change.

When travelling to remote locations, it is often helpful to bring cash, in case you want to buy something from someone who does not accept credit or debit cards. It is also helpful to bring a variety of denominations in case the seller does not have change. Even so, you may not have the exact amount, and will have to pay a little bit more than full price. The same problem can arise even in urban locations, for example with vending machines that do not return change.

Of course, you would like to minimize the amount you pay (though you must pay at least as much as the value of the item). Moreover, while paying the minimum amount, you would like to minimize the number of coins or bills that you pay out.

Input Specification

You have to read the input from a text file "ExactChange.txt". The first line of input contains one integer specifying the number of test cases to follow. Each test case begins with a line containing an integer, the price in cents of the item you would like to buy. The price will not exceed 10 000 cents (i.e., $100). The following line contains a single integer n, the number of bills and coins that you have. The number n is at most 100. The following n lines each contain one integer, the value in cents of each bill or coin that you have. Note that the denominations can be any number of cents; they are not limited to the values of coins and bills that we usually use in Canada. However, no bill or coin will have a value greater than 10 000 cents ($100). The total value of your bills and coins will always be equal to or greater than the price of the item.

Sample Input

```
1
1400
3
500
1000
2000
```

Output Specification

For each test case, output a single line containing two integers: the total amount paid (in cents), and the total number of coins and bills used.

Output for Sample Input

```
1500 2
```

Work Reduction

Paperwork is beginning to pile up on your desk, and tensions at the workplace are starting to mount. Your boss has threatened to fire you if you don't make any progress by the end of the day. You currently have **N** units of paperwork on your desk, and your boss demands that you have exactly **M** units of paperwork left by the end of the day.

The only hope for you now is to hire help. There are various agencies which offer paperwork reduction plans:

For **$A** they will reduce your paperwork by one unit.
For **$B** they will reduce your entire paperwork by half (rounding down when necessary).

Note that work can never be reduced to less than 0.

Your task now is to produce a sorted table of agency names and their respective minimum costs to solve your workload problem.

You have to read the input from a text file "WorkReduction.txt" .The first line of input consists of a single positive integer representing the number of cases to follow. Each case begins with three positive integers separated by spaces: **N** - your starting workload, **M** - your target workload, and **L** - the number of work reduction agencies available to you, ($1 <= M <= N <= 100000$, $1 <= L <= 100$). The next **L** lines have the format "[*agency name*]**:A,B**", where **A** and **B** are the rates as described above for the given agency. ($0 <= A,B <= 10000$) The length of the agency name will be between 1 and 16, and will consist only of capital letters. Agency names will be unique.

For each test case, print "Case X", with X being the case number, on a single line, followed by the table of agency names and their respective minimum costs, sorted in non-decreasing order of minimum costs. Sort job agencies with identical minimum costs in alphabetical order by agency name. For each line of the table, print out the agency name, followed by a space, followed by the minimum required cost for that agency to solve your problem.

Sample Input

```
2
100 5 3
A:1,10
B:2,5
C:3,1
1123 1122 5
B:50,300
A:1,1000
C:10,10
```

D:1,50
E:0,0

Sample Output

Case 1
C 7
B 22
A 37
Case 2
E 0
A 1
D 1
C 10
B 50

Solved.

Perfect P*th* Powers

We say that x is a perfect square if, for some integer b, $x = b^2$. Similarly, x is a perfect cube if, for some integer$b$, $x = b^3$. More generally, x is a perfect p*th* power if, for some integer b, $x = b^p$. Given an integer x you are to determine the largest p such that x is a perfect p*th* power.

You have to read the input from a text file "PerfectPthPowers.txt". Each test case is given by a line of input containing x. The value of x will have magnitude at least 2 and be within the range of a (32-bit) *int* in C, C++, C# and Java. A line containing 0 follows the last test case.

For each test case, output a line giving the largest integer p such that x is a perfect p*th* power.

Sample Input

```
17
1073741824
25
0
```

Output for Sample Input

```
1
30
2
```