1 Seconds To Go
Input File: SecondsToGoIn.txt

Nadia is very excited to be returning home after being away for two months. The pilot of the airplane she is on has just announced the hours and minutes remaining in the flight, and she would like to begin counting down the seconds until touchdown. Your task is to convert the remaining hours and minutes of flight time to seconds, so that she can begin her countdown.

Inputs:
There will be one line of input that contains two integers separated by a space. These represent the number of hours, followed by the number of minutes, until the airplane lands.

Outputs
There will be one integer output: the number of seconds remaining in Nadia's flight, annotated as shown below.

Sample Input	**Sample output**
`1 10`	`4200 seconds to go`

2 Seven Times Tables
Input File: SevenTimesIn.txt

Bob has been asked to identify which numbers in a given sequence of numbers, each of which are less than one billion, are members of the seven times table. Although he memorized the table in grammar school: 7, 14, 21, 28, 35, 42, … he was not required to go past 70. Therefore, he has asked you to write a program to identify the members of the seven times table that appear in the sequence of numbers he will be given.

Inputs:
The first line of input will contain an integer that is the number of sequences Bob will be asked to examine. This will be followed by one line of input per sequence that contains integers, each separated by a space. The first integer on a line will be a count of the numbers in the sequence, and the remaining integers on the line will be the numbers that make up the sequence.

Outputs
There will be one line of output per sequence that contains the numbers in the sequence that are part of the seven times table.

Sample Input	**Sample output**
2 6 32 70 62 147 626326 10326722 4 41132 61334 712349 98765436	70 147 10326722 41132 61334 98765436

3 Ronnie's Ribbons
Input File: RonniesIn.txt

Ronnie has collected pieces of red ribbons of various lengths. When she needs a length of ribbon that she does not have, she locates *two* pieces of ribbon from her collection whose combined length is the length she needs. Then she tapes them together end-to-end (with no overlap).

Your task is to determine how many ribbons of a given length she could produce from her collection of ribbon pieces, given the length of ribbon she needs and the length of each piece of ribbon in the collection.

Inputs:
The first line of input will be the number of ribbon piece collections to consider, followed by two lines of input per collection. The first of these two lines contains two integers that represent the length of ribbon Ronnie needs follow by the number of ribbon pieces, **n**, in the collection. The second line will contain **n** integers that represent the length of each of the ribbon pieces in the collection. The inputs will be separated by a space.

Outputs
There will be one output per ribbon collection that represents the number of ribbons of the given length she could produce from the collection.

Sample Input

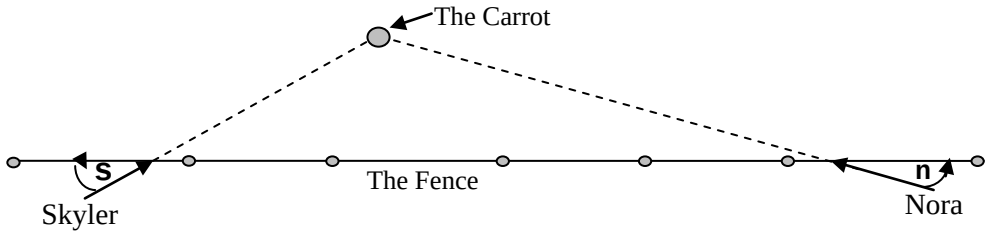
```
2
7 6
3 4 3 6 1 1
20 10
2 5 18 15 2 15 16 30 6 14
```

Sample output

```
2
3
```

4 Running Rabbits
Input File: RunningIn.txt

Two hungry rabbits, named Skyler and Nora, are running in a straight line through a field towards a carrot. Skyler is approaching it from the left, and Nora is approaching it from the right as shown below. On their way to the carrot, they pass under a fence at *exactly* the same time. Skylers's path is inclined to the fence at **s** degrees, and Nora's path is inclined to the fence at **n** degrees. They both run at a constant, but most times different, speed and they want to arrive at the carrot at the same time.



Given Skyler's speed and the inclination of the rabbits' running directions, your task is to compute Nora's speed such that they will arrive at the carrot at the same time.

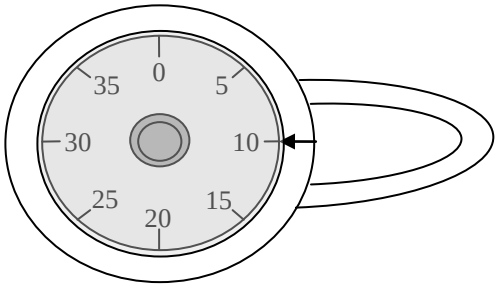
Inputs:
The first line of input will be the number of cases to consider, followed by one line of input per case containing three integers each separated by a space. The first of these integers will represent Skyler's running speed. The second integer will represent Skyler's running angle relative to the fence, **s**, and the third integer will represent Nora's running angle relative to the fence, **n**. The units of both angles is degrees.

Outputs
There will be one output line per case that represents Nora's speed such that Nora and Skyler will arrive at the carrot at the same time. All output should be expressed with one digit of precision rounded to the nearest tenth.

Sample Input	**Sample output**
3	
30 45 45	30.0
10 60 10	49.9
20 30 60	11.5

5 Lock Ticks
Input File: LockTicksIn.txt

Logan, being always pressed for time, wants to purchase a standard combination lock that opens with the least number of “ticks”. The ticks are numbered from 0 to 39 on the rotatable dial (center portion) of the lock, increasing in the clockwise direction. The fixed part of the lock has an arrow on it, which always "points to" one of the ticks on the dial. Of course, the arrow points to different ticks as the dial is turned.



The lock comes with three code numbers T1, T2, T3. These are non-negative integers less than 40, and no two of the three are the same. The lock is opened using a three step procedure:

1. Turn the dial clockwise exactly two full revolutions, and continue to turn it clockwise until the arrow points to tick T1.
2. Turn the dial one full revolution counterclockwise and continue to turn it counterclockwise until the arrow points to tick T2.
3. Turn the dial clockwise until the arrow points to tick T3.

The lock should then open. Your task is to tell Logan how many ticks are required to open a lock, given the lock’s starting tick, and the lock’s combination: T1, T2, T3.

Inputs:
There will be one line of input that contains the number of locks Logan is considering. This will be followed by one line per lock that contains four integers each separated by a space. These integers represent the starting tick, followed by the lock’s code numbers: T1, T2, T3.

Output:
There will be one line of output for each lock that contains the number of ticks required to open the lock.

Sample input

```
6
0 30 0 35
9 19 6 32
35 0 10 5
0 39 0 38
4 5 6 7
7 6 5 4
```

Sample output

145
191
170
124
199
161

6 Word Worm
Input File: WordWormIn.txt

Walter, like other members of his Word Worm species, crawls around rectangular grids of text looking for hidden words. One such grid is shown below. He can crawl either straight to the right, left, up, or down, or along any of the four diagonal directions: up and to the right, down and to the right, up and to the left, or down and to the left. When he finds a word along one of these eight directions, he records the word in his journal, and then crawls along the letters of the word from its first letter to its last. As he does this, he also enters the row and column numbers of his path along the word into his journal.

```
Column number → 0 1 2 3 4 5 6 7 8 9
                0 R T W W I L N P Z C
                1 I E K X L M J E I O
                2 P Q T S N A G O L O
                3 V X P U F I W C V L
                4 T O P L P Q E R D S
                5 H I I I E M X S J H
                6 N O M W E Q O S U S
                7 M C P E U I C C P I
   Row number ↓
```

For example, as he crawled around the above grid, one of the entries he would make in his journal is PIE (4, 2) (5, 3) (6, 4). Your task is to produce the contents of Walters's journal, for a given rectangular grid of text.

Inputs:
The first line of input will be the number rectangular grids of text to consider, followed by one group of inputs for each grid. The first line in a group will contain the words to find, each separated by a space. The second line of will contain one integer that is the number of rows in the rectangular text grid, **r**. This will be followed by **r** lines that contain the letters in each row of the grid, with each letter on a line separated by a space.

Outputs:
There will be one line of output per word searched for. It will contain the word being searched for followed by the text: **Not Found** if the word is not found. Otherwise the word being searched for will be followed by the row and column location of the first letter of the word, followed by the row and column location of the second letter of the word, etc. The row and column numbers will be output as: (row, column) with each row and column number output separated by a space.

(Sample inputs and outputs are on the next page)

Sample Input

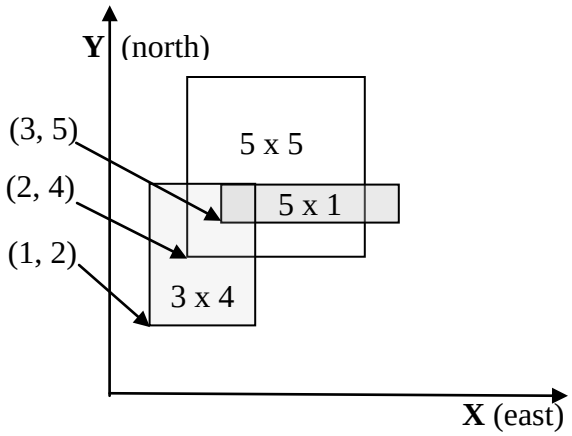
```
2
COMPUTER LOGAN EGG PIE HI COOL TO
8
R T W W I L N P Z C
I E K X L M J E I O
P Q T S N A G O L O
T X P U F I W C V L
T O P L P Q E R D S
H I I I E M X S J H
N O M W E Q O S U S
M C P E U I C C P I
HELP HEAD WALK
4
H O L W
H E A M
T L L C
K H A P
```

Sample Output

```
COMPUTER (7, 7) (6, 6) (5, 5) (4, 4) (3, 3) (2, 2) (1, 1) (0, 0)
LOGAN (2, 8) (2, 7) (2, 6) (2, 5) (2, 4)
EGG NOT FOUND
PIE (4, 2) (5, 3) (6, 4)
HI (5, 0) (5, 1)
COOL (0, 9) (1, 9) (2, 9) (3, 9)
TO (4, 0) (4, 1)
HELP (0, 0) (1, 1) (2, 2) (3, 3)
HEAD NOT FOUND
WALK (0, 3) (1, 2) (2, 1) (3, 0)
```

7 Coverage
Input File: CoverageIn.txt

Old McDonald has a farm, and has hired a firm to partially cover several of his planting fields with rectangular tarpaulins to reduce weed growth. He has specified that the width and height of the tarpaulins run east-west and north-south respectively. After completing the task, the workers realized that some of the tarpaulins overlap, and the firm agreed to only charge Mr. McDonald for the *net* area covered. That is, they will only charge him once for areas of the field covered by two or more tarpaulins. For example, Mr. McDonald would be charged for 34 square feet of *net* coverage for the three tarpaulins positioned as shown below.



Your task is to determine the net coverage and net cost of the tarpaulin installation on each planting field given the locations and sizes of the tarpaulins, and the cost per square foot of net coverage.

Inputs:
The first line of input will be the number of planting fields to consider, **p**, followed by one group of inputs for each field. The first line in a grouping will contain an integer, **n**, that represents the number of tarpaulins placed in the field, followed by an integer that represents the cost of the tarpaulin installation per *net* square foot. This will be followed by **n** lines of input, one line per tarpaulin that contains four integers. These integers represent the width of the tarpaulin, followed by its height, followed by the x and y coordinates of its lower left corner. The units of coordinate system and the tarpaulin sizes is feet, and all inputs on a line are separated by a space.

Outputs:
There will be one line of output per planting field that contains two integers separated by a space. The first number will be the net area of the planting field covered by the tarpaulins, and the second number will be the cost of the installation.

(Sample inputs and outputs on next page)

Sample Inputs

```
2
3 100
3 4 1 2
5 5 2 4
5 1 3 5
5 200
5 5 2 1
5 2 0 2
4 5 1 0
3 4 3 3
1 2 0 1
```

Sample Outputs:

```
34 3400
39 7800
```