MERCER UNIVERSITY

Spring Programming Contest

February 29, 2020

An Hour Away	(prob1)
Geospatial Coverage	(prob2)
Leap Day	(prob3)
Leap Day Dinner	(prob4)
Party	(prob5)
Pigeon-Hole Proofs	(prob6)
Price is Right	(prob7)
Root Probability	(prob8)
Sidekicks	(prob9)
Which Side of the Office	(prob10)
Motif Finding	(prob11)

The name in parenthesis following each problem is the name you must use as your program's name. You must add the appropriate extension depending upon your choice of programming language (.c .cpp .java .py).

An Hour Away (prob1)

The Problem

You have planned an exciting vacation to an amazing location far, far, away. You are very anxious to get to your destination as fast as possible. Your flight just landed at the airport nearest to your destination, and you now need to rent a car to finish your journey. At the car rental kiosk, you are informed that they only rent “Smart Self-Driving Cars.” Being the techy person that you are, you are very excited about riding in this very special car. The rental agent then informs you that their smart cars have an unusual software glitch. When you enter your destination in the smart car, it calculates the shortest distance, and begins driving. The unusual part is that the car continuously adjusts its speed to always be “an hour away” from your destination. That is, when you are 100 miles away, the car travels 100 mph; when you are 97.5 miles away the car travels 97.5 mph; when you are 1 mile away the car travels 1 mph.

Being the great technical problem solver that you are, you realize at this rate, you will never get to your destination! You also know that you can stop the car at any point and walk. You, therefore, must determine when to stop the car and start walking in order to get to your amazing vacation the fastest.

Input

Your input will begin with a single line containing the number of vacation test cases $1 <= n <= 20000$. The following n lines will each contain a test case that consists of 2 integers: $1 <= d <= 1000$, and $1 <= s <= 20$, where d is the distance in miles from the airport to the vacation spot, and s is your constant walking speed in miles per hour.

Output

Your output should, for each test case, be a single floating point number denoting the least amount of time in hours needed for each vacation test case. Output should be rounded to the nearest hundredth of an hour.

Sample Input

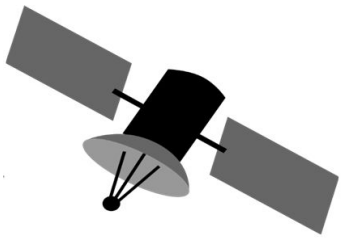
```
4
100 3
50 3
8 5
1000 1
```

Sample Output

```
4.51
3.81
1.47
7.91
```

Geospatial Coverage (prob2)

The Problem



Little Bobby Tables finally grew up and started his own satellite image mapping company. They have been extremely successful despite one major problem, not knowing how much of the earth they've mapped. Oh well, baby steps. You're a new developer with the company, and you've received the unenviable task of finding the answer.

Input

Input begins with a single line containing a single number *n* (<= 10), the number of test cases. Each of the subsequent *n* sections begins with three numbers separated by a single space: *x y z*. *x* (<= 1000) is the width of the area to be mapped, and *y* (<= 1000) is the height. *z* (<= 10) is the number of satellites that will be used to map the area. (Note: the area is completely flat and fully rectangular). *z* lines of input will follow. Each line will contain 5 numbers separated by a single space: *x y j k l*. *x, y* are coordinate points which indicate where the satellite enters the area. *j, k* are coordinate points which indicate where the satellite leaves the area and *l* is the scanning width. (Note: the coordinate pairs are guaranteed to be along the outside edges of the grid).

Note: Scanning width is defined differently than you might expect. For example, if a satellite enters the area at 0, 0 with a scanning width of 10, the satellite's scan would stretch to 0, 10 and 10, 0. If the satellite enters at 0, 5 with a scanning width of 10, the scan would stretch to 0, 15 and 5, 0.

Output

Output the percentage of area covered by the satellites, rounded to 0 decimal places, as shown in the sample output.

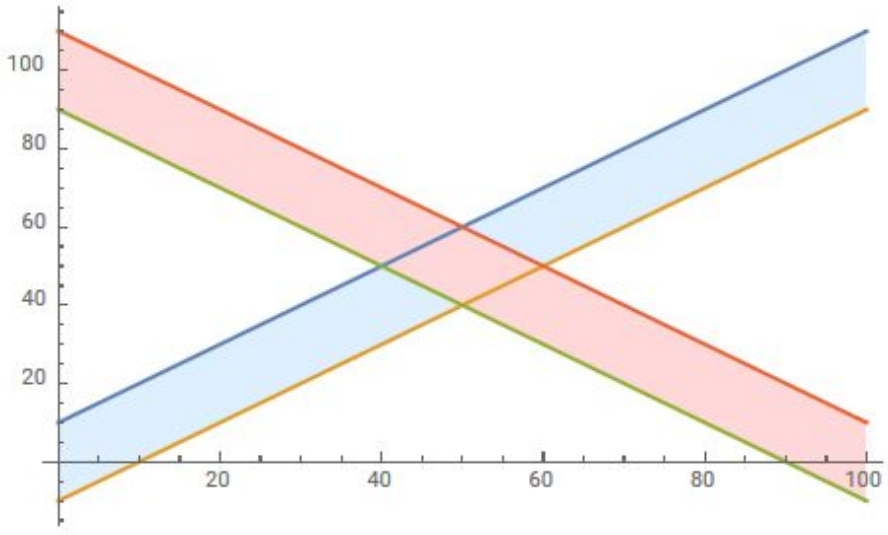
Sample Input

```
2
100 100 2
0 0 100 100 10
0 100 100 0 10
50 50 2
0 25 50 25 5
25 0 25 50 5
```

Sample Output

```
36.00
36.00
```

Explanation of Sample Output 1



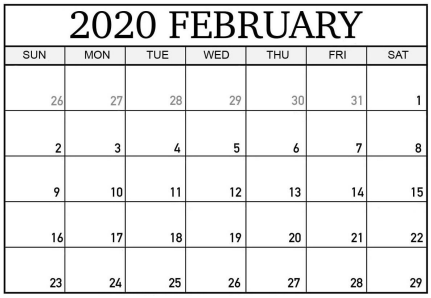
The intersection of the polygons in the middle is 200 square feet. That area is counted only once. The total area for one polygon 1900. Thus the total scanned area is 3600 ft. Dividing by 10000 yields 0.36 or 36 percent rounded.

Note: Only space within the defined region is counted.

Leap Day (prob3)

The Problem

2020 is a leap year, and today is our extra leap day in February. We want to write a computer program which will calculate the total leap years in an inclusive range of years. As you may know, this extra day keeps our Gregorian calendar year from drifting through the seasons over time since a year is technically 365.25 days. Here is the rule.



2020 FEBRUARY

SUN	MON	TUE	WED	THU	FRI	SAT
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

> Every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. For example, the years 1700, 1800, and 1900 are not leap years, but the years 1600 and 2000 are.

Input

The input will start with a single line containing a positive integer *n* representing the specified number of year ranges to be input. Each of the *n* line(s) consists of two integers starting with the first year, a single space, and followed by the second year. You may assume all years fall in the range [1600, 2100], and that the second year is always larger than the first year.

Output

Keep track of all line numbers and output the code for each line in the exact format below. If there is only one leap year, use the word “is” rather than “are” in your output along with the singular “year”. Make sure you end the output of each line with a period.

Sample Input

```
5
2015 2016
2015 2020
2014 2015
1900 2000
1900 2100
```

Sample Output

```
There is 1 leap year from 2015 to 2016.
There are 2 leap years from 2015 to 2020.
There are 0 leap years from 2014 to 2015.
There are 25 leap years from 1900 to 2000.
There are 49 leap years from 1900 to 2100.
```

Leap Day Dinner (prob4)

The Problem

Ah, Leap Day. The traditional rhubarb-flavored caramels, the presents wrapped in yellow and blue, the city-wide ice maze, and the huge Leap Day extended family feast!

Your family is hosting the feast this year. As per tradition, dating as far back as 1996, the hosting family presents a list of dishes to the invitees. The invitees send back votes on which dishes they would most enjoy. Based on arcane rules and a two-hour meeting, the hosts pick a threshold vote value to decide the menu. A set of dishes from the list are picked such that the sum of all votes for those dishes is equal to or greater than the chosen threshold.

Cooking all those dishes is a lot of work, and your family is under a lot of pressure to decide the menu and get it prepared by the big day. You despise cooking, so you offer a bargain: given the list of dishes, the total votes for each, and how long each dish takes to prepare, you'll figure out a set of dishes that can be prepared in the least amount of time and still meet or exceed the vote threshold. If you can do that, you'll be excused from cooking duties.

Note that despite multiple people in the kitchen, only one dish can be in progress at a time.

Input

The first line of input will contain a single positive integer, ***c*** (***c*** ≤ 10), representing the number of input cases to process. The input cases follow. The first line of each case has two integers: ***n*** (1 ≤ ***n*** ≤ 5,000), the number of dishes; and **t** (1 ≤ ***t*** ≤ 1,000,000), the target threshold value. ***n*** lines follow, each describing one dish. Each dish line has two integers: ***m*** (1 ≤ ***m*** ≤ 10), the number of minutes the dish takes to prepare, and ***v*** (1 ≤ ***v*** ≤ 1,000), the number of votes for the dish.

Output

For each input case, on a line by itself, output the minimum number of minutes in which a conforming set of dishes can be prepared.

Sample Input

```
2
2 10
1 1
1 9
4 20
2 9
3 10
2 5
3 5
```

Sample Output

```
2
7
```

Party (prob5)

The Problem

You plan to drive to pick up all your friends for an upcoming party you are hosting. You and your friends live in different houses in the same city and each house has an index. The roads guarantee that there exists a path between every two houses. Starting from your house, calculate the shortest path for you to pick up every friend and finally return to your home. Assume your car can carry all friends at the same time and you live in the house indexed as 1.

Input

The first line of input will contain a single integer t, which represents the number of test cases. You may assume that $1 \leq t \leq 100$. Each test case always consists of at least three lines in the following format.

The first line of each test case consists of three integers n, m, and k. Assume n is a positive integer, $1 \leq n \leq 50{,}000$, representing the number of houses and each house has a unique index from 1 to n. This is followed by a positive integer m, $1 \leq m \leq 100{,}000$, representing a number of roads, and k is an integer, $0 \leq k \leq 15$, representing your number of friends.

The second line of each test case consists of k integers, representing your friends' house numbers. This is followed by m lines for each of your roads. For each road, you will have three integers $h1$, $h2$, and w, showing that a two-way road with length w is connecting houses $h1$ and $h2$. Assume w is an integer where $0 \leq w \leq 50000$.

Output

An integer, which is the length of the shortest path you will take.

Sample Input

```
1
4 5 2
3 4
1 2 1
2 3 3
3 4 2
1 4 3
1 3 5
```

Sample Output

```
9
```

Pigeon-Hole Proofs (prob6)

The Problem

Freddie learned about the pigeon-hole principle in Discrete Math class. One of the homework problems, an even-numbered one of course, is to prove that for every modulus *M*, there exists a positive number *K*, having only 3's and 0's in its decimal representation, such that *K* mod *M* is 0.

Your task is to help Freddie quickly find the smallest *K* for various values of *M*. Your submission should be able to find a solution per test case each second.

Input

The input will consist of a number *N*, $1 \leq N \leq 60$, which is the number of lines/test cases. Each of the following *N* lines will contain one integer *M*, $1 \leq M \leq 2{,}000{,}000$.

Output

For each test case, output the smallest (positive) value *K*, having only 3's and 0's in its decimal representation, such that *K* mod *M* = 0.

Sample Input

```
3
1
5
27
```

Sample Output

```
3
30
333333333
```

Price is Right (prob7)

The Problem

In the TV Game Show "The Price is Right", an item is described to four contestants and then each contestant guesses the price of the item. The contestant closest to the actual price of the item (without going over) wins the item. The four guesses are unique so there will not be ties. However, if all four contestants guess over the actual price, no one wins in that round. You are to write a program to play this game.

Input

The first input line will contain a single positive integer, ***g*** (***g*** $\leq$ 50), representing the number of input games to play. The games are on the following ***g*** input lines, one game per line. Each game consists of five positive integers: the actual price followed by the four distinct guesses (by the contestants). Each of these integers will be less than or equal to 10,000.

Output

For each input game, output the contestant number who wins the game. If all four contestants are over the actual price, print 0 (zero).

Sample Input

```
4
500 550 400 300 350
1000 800 999 1010 1000
50 600 70 80 900
700 850 950 1 2000
```

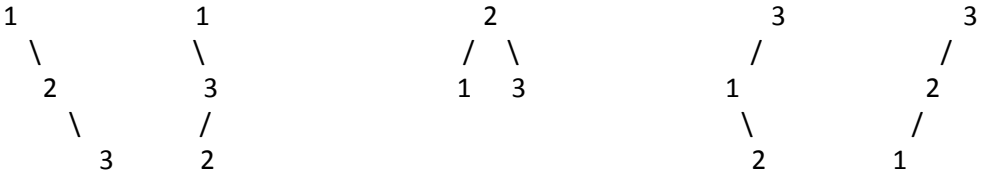
Sample Output

```
2
4
0
3
```

Root Probability (prob8)

The Problem

Given the integers 1 through n, there are several different ways to arrange those integers into a valid binary search tree. For example, for n = 3, there are five possible ways:



As we can see, not all roots are equally likely. For example, if we choose one of these five binary search trees randomly, the probability the root is 1 is 2/5 while the probability the root is 2 is only 1/5. Given a positive integer, n, and a potential root value k, determine the probability that a randomly selected binary search tree storing the integers 1 through n, inclusive, has k as its root. (Notice that for some orderings of inserting values into the tree, the tree takes different structures, but not always. For the purposes of this problem, we assume each unique possible structure of the binary search tree is equally likely.)

Input

The first line of input will contain a single positive integer, ***c*** (***c*** $\leq$ 500), representing the number of input cases to process. The input cases follow, one per line. Each input case has two positive integers, ***n*** (***n*** $\leq$ 30), representing the number of nodes in the binary search tree and ***k*** (***k*** $\leq$ ***n***), representing the potential root of the tree.

Output

For each input case on a line by itself, output the probability that k is the root of a randomly selected tree. Present the output as a fraction in lowest terms. Specifically, print the output in the form p/q, where p and q are positive integers with gcd(p, q) = 1, where the fraction p/q is the desired probability.

Sample Input

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

Sample Output

```
2/5
1/5
1/7
```

Sidekicks (prob9)

The Problem

You are the world's greatest detective. You have taken down some serious bad guys using the skills you learned in the mountains. Now a new bad guy, Half and Half Man, the dairy fiend, has captured Lieutenant Morgan's family, and you need to stop him. Half and Half Man has hired many thugs to keep his hostages captured. To save everyone you will use your sidekicks to disable all the thugs.

Each thug has some strengths and weaknesses that your sidekicks might be able to exploit. Not all sidekicks have the same abilities. Some sidekicks might not be able to defeat every thug. Additionally, each sidekick has some required amount of time to take down any given thug. You want to rescue the Lieutenant's family as quickly as possible. Determine the earliest time in which your sidekicks can disable all thugs.

Input

Input will begin with a single integer, t ($t \leq 25$), representing the number of test cases. The remainder of the input will consist of t case descriptions. Each case description will begin with 2 space separated, positive integers, n and m ($1 \leq n, m \leq 200$), representing the number of sidekicks and the number of hired thugs respectively. The case description will end with n lines each containing one sidekick description. Each sidekick description will begin with 2 space separated, positive integers, r_i and p_i ($r_i \leq 1000$, $p_i \leq m$), representing how long it takes the given sidekick to research a method to defeat any thug they are capable of defeating and the number of thugs they are capable of defeating. Following this the sidekick description will contain p_i distinct, space separated, positive integers, a_{ij} ($a_{ij} \leq m$), representing the thug ids the given sidekick is capable of defeating. The thugs are labeled 1 to m, but the list of a_{ij} may not be in numerical order. You are guaranteed that each thug is defeated by at least one sidekick.

Output

For each case, on a line by itself, output the smallest amount of time necessary for all thugs to be defeated.

Sample Input

```
3
2 2
2 2 1 2
6 2 1 2
5 5
1 1 1
2 1 2
3 1 3
4 1 4
1000 1 5
2 5
5 3 1 2 4
2 3 1 3 5
```

Sample Output

```
4
1000
10
```

Which Side of the Office? (prob10)

The Problem

Will and Melina share an office that is the shape of a convex polygon. In order to divide the space, they have agreed upon the following rules: Will will choose two distinct vertices of the polygon and a line will be drawn between those two vertices, forming two separate polygons. Melina will choose the larger of the two polygons (in area) and Will is left with the other polygon, for his side of the office. Naturally, Will wants to maximize the area of his side of the office. Help him figure out the maximum possible area of his side of the office if he chooses his vertices appropriately.

Input

The first line of input will contain a single positive integer, c ($c \leq 100$), representing the number of input cases to process. The input cases follow. The first line of each input case has a single positive integer, n ($4 \leq n \leq 20$), representing the number of vertices of the polygon representing the office space. The following n lines contain the vertices, one per line, in clockwise order. Specifically, the i^{th} ($1 \leq i \leq n$) of these lines contains two space separated integers, x_i and y_i, ($|x_i| \leq 1000$, ($|y_i| \leq 1000$) respectively, where (x_i, y_i) represents the i^{th} point of the polygon in clockwise order.

Output

For each input case, on a line by itself, output an integer representing **twice the maximum possible area** that Will's side of the office could be.

Sample Input

```
2
4
0 1
1 1
1 0
0 0
5
-10 10
0 20
10 10
5 0
-5 0
```

Sample Output

```
1
200
```

Motif finding problem (prob11)

The Problem

Consider a set of n l-bit integers: $\{S_1, S_2, \ldots, S_n\}$. Given a number d, where $d < l$, the goal is to find all integers which are at a Hamming distance of at most d from all of the given n integers. A Hamming distance between two sequences (in this case, bit representation of integers) of equal lengths is the number of corresponding positions where the sequences differ.

Input

The first line of input contains a single positive integer, c ($c \leq 25$), representing the number of input cases. The input cases follow. The first line of each input case contains three space separated positive integers, n ($2 \leq n \leq 100$), l ($3 \leq l \leq 30$), and d ($d < \min(10, l)$), respectively. The following n lines each contain one distinct l-bit integer, written as an unsigned base ten integer.

Output

For each input case, start with a single line with the following format:

```
Case X: Matching Motifs
```

Where X is the case number starting with one. The following lines should have the solutions for the case, sorted in numerical order, one per line. Follow the output for each case with a new line. The cases have been constructed such that each case has in between 1 and 200 solutions, inclusive.

Sample Input

```
2
5 24 6
7044851
2179191
7098723
4294775
6768819
2 3 1
5
7
```

Sample Output

```
Case 1: Matching Motifs
6379635
6899827
6903923

Case 2: Matching Motifs
5
7
```