2015 FHSPS Playoff
May 16, 2015
Online

Filename	Problem Name
eye	Top of the Eye
lasertag	Laser Tag
movie	Dinner and a Movie
radio	Radio Prizes
science	Science Center Membership
soccer	Orlando City Soccer
speed	Theme Park Speed
swan	Swan Boats

Top of the Eye

Filename: eye

The World's largest ferris wheel is in Orlando and you can't wait to visit. Unfortunately, there's a nefarious kid who drops pennies from various points on the wheel. It's not safe to visit until you write a program to calculate where those pennies will land, so that other guests can avoid them!

Assume that the top of the ferris wheel is at an elevation of 405 feet off the ground and the bottom of the ferris wheel is at an elevation of 5 feet off the ground, thus the radius of the ferris wheel is 200 feet. If we view the ferris wheel from afar, let the y-axis be vertical and the x-axis be horizontal to the ground with the center of the ferris wheel at (0, 205). Define θ to be 0 degrees when a passenger is seated at (200, 205), and let *v* be the velocity of the wheel, which for the purposes of this problem, will be assumed to be constant. Assume that the wheel spins counter-clockwise. Thus, in terms of θ and *v*, our instantaneous horizontal (v_x) and vertical (v_y) velocities are as follows:

$$v_x = -v\sin\theta$$
$$v_y = v\cos\theta$$

Ignore the effects of air resistance on the penny for the purposes of this problem. Thus, if a penny is given a horizontal velocity of *v*, its change in x over time *t* is *vt*. If a penny is given a vertical velocity of *v*, its change in y over time *t* is $vt - \frac{1}{2}at^2$, where *a* is the acceleration due to gravity. For this problem, use $a = -32ft/sec$.

The Problem

Given the angular velocity of the ferris wheel, as well as the current position (given as an angle) of our nefarious penny dropper, determine the x-coordinate, as defined in the problem description above, that the penny will hit the ground.

The Input

The first line of the input file will contain a single positive integer, *n* ($n \leq 1000$), representing the test cases to process. The test cases will follow, one per line. Each test case will have a positive real number *a* ($0.05 \leq a \leq 10$), representing the angular velocity in degrees per second, to at most 2 decimal places, followed by a space, followed by a non-negative integer, *p* ($p < 360$), representing the angular position of the penny dropper on the ferris wheel at the time that he releases his penny.

The Output

Output a single integer on a line by itself representing the rounded value of the x-coordinate of where the penny will land. It is guaranteed that the fractional part of each actual x-coordinate will not be in the range [.499, .501].

Sample Input

```
2
1.0 270
5.0 0
```

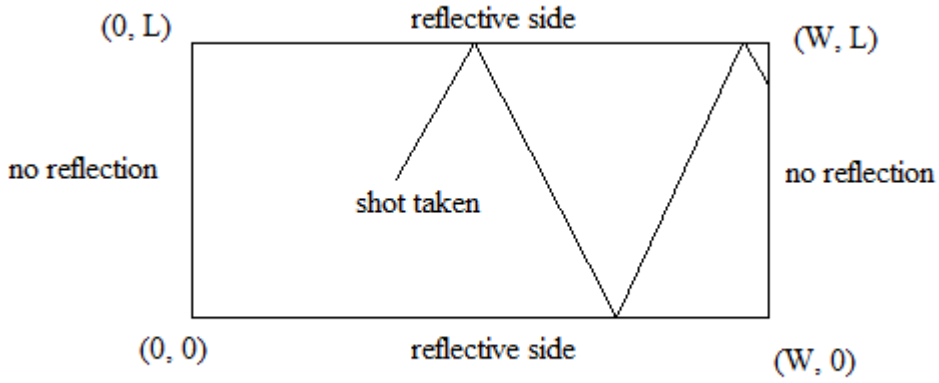
Sample Output

```
2
200
```

Laser Tag

Filename: lasertag

You love going to play laser tag with your friends. The new laser tag arena in town features a rectangular playing field with two of the parallel sides that perfectly reflect laser shots! This is great for people like you, who aren't great shots. Even if the initial trajectory isn't correct, after a few bounces, your shot may hit someone. Consider the diagram below:



We model the laser tag arena as a rectangle in the first quadrant, with corners at (0, 0), (W, 0), (W, L) and (0, L), in counter-clockwise order. A shot taken from the position shown at the angle shown, will bounce perfectly off both sides that are parallel to the x-axis using the given coordinate system. Once a shot makes contact with either of the vertical sides, it gets absorbed into that material and will no longer be reflected.

You would like to figure out different angles at which to shoot so that you can hit your opponent, so that you have multiple options. For the purposes of this problem, we assume that both you and your opponent are points and that reflections of a laser from a shot off both reflective sides follow the law of reflection.

The Problem

Given your position in the laser tag arena, your opponent's position, and a range of angles you are able to shoot (with 0^o representing shooting in the positive x-axis and 90^o representing the positive y-axis) within, determine the number of different angles you can shoot at and still hit your opponent.

The Input

The first line of the input file will contain a single positive integer, n ($n \leq 50$), representing the test cases to process. The test cases will follow. The first line of each test case has two space separate positive integer, W ($W \leq 1000$), and L ($L \leq 1000$), representing the width and length, respectively, in feet of the laser tag arena. The second line of each test case has two space separate positive integers, x_1 ($x_1 < W$), and y_1 ($y_1 < L$), representing your (the shooter's) x and y position in the arena. The third line of each test case has two space separate positive integers, x_2

($x_1 < x_2 < W$), and *y_1 ($y_1 < L$)*, representing your opponent's position in the arena. The fourth line of each test case will have two integers, *A_{low} ($-90 < A_{low} < 90$)* and *A_{high} ($A_{low} < A_{high} < 90$)*, representing the range of angles from which you are able to shoot. Note: None range boundaries given will correspond to a shot that hits the target exactly. Namely, a shot directed at exactly angle *A_{low}* will not hit the target and a shot directed exactly at angle *A_{high}* will not hit the target either.

The Output

For each test case, output a single integer on a line by itself representing the number of different angles at which you can aim your shot within the given range to hit your opponent.

Sample Input

```
2
1000 500
50 250
950 250
30 40
1000 100
50 50
500 10
-60 40
```

Sample Output

```
0
12
```

Dinner and a Movie

Filename: movie

The downtown movie theater, which serves food during their movies, is running a special for large groups. Any group that buys 10 or more movie tickets can buy slices of pizza for $1 each. Normally, slices are $2 each. Note that movie tickets cost $10 each. Since the movie theater's cash register is having some trouble ringing up this special, they've asked you to write a small program to calculate the total cost for each group buying tickets and pizza.

The Problem

Given the number of movie tickets and slices of pizza a group wants to buy, determine their total cost, in dollars.

The Input

The first line of the input file will contain a single positive integer, n ($n \leq 1000$), representing the number of test cases to process. Each of the test cases will follow, one per line. Each test case will have two space separated positive integers, m ($m \leq 1000$) and p ($p \leq 10000$), representing the number of movie tickets and number of slices of pizza bought by the corresponding group.

The Output

Output a single integer on a line by itself for each test case indicating the cost for the corresponding group, in dollars.

Sample Input

```
3
5 2
10 20
100 999
```

Sample Output

```
54
120
1999
```

Radio Prizes

Filename: radio

A local radio station offers various cash prizes on various days. However, once you win a prize, you have to wait a certain number of days before you can win another prize. You have figured out how to call faster than all of the other callers so that you can win any prize offered and you have gotten your hands on the complete schedule of prizes, including the day a prize is offered, its cash value, and the number of days you would have to wait until being allowed to win another prize. Your goal is to determine the most amount of money you could win.

For example, if the prizes were as follows:

Day	Value ($)	Wait (in days)
2	80	9
8	50	2
10	40	2
13	20	5

You could take the prize on days 8, 10 and 13 for a total cash value of $110. Notice that if you take the prize on day 2, you can't claim another prize until day 11, at which point only the $20 prize on day 13 is left, leaving you with a total cash value of $100. Thus, for this problem instance, the correct answer (maximal possible cash winnings) is $110.

Since the radio station has limited resources, it never offers more than one prize per day.

The Problem

Given a complete radio station prize schedule, determine the maximum amount of money you could win under the rules.

The Input

The first line of the input file will contain a single positive integer, n ($n \leq 100$), representing the number of test cases to process. Each of the test cases will follow. The first line of each test case will have a single positive integer, k ($k \leq 1000$), representing the number of prizes the radio station will offer. The i^{th} ($1 \leq i \leq k$) line that follows will contain three space separated integers: d_i ($1 \leq d_i \leq 10000$), v_i ($1 \leq v_i \leq 1000000$), and w_i ($1 \leq w_i \leq 10000$), representing the day, value and wait time afterwards of the i^{th} prize. Note that the prizes may not be listed in chronological order.

The Output

Output a single integer on a line by itself for each test case indicating the maximum value in prizes you can collect following the rules of the radio station, in dollars.

Sample Input

```
2
4
2 80 9
8 50 2
10 40 2
13 20 5
3
10 100 9
20 200 10
30 300 11
```

Sample Output

```
110
600
```

Science Center Membership

Filename: science

One of the better museums in town is the science museum. Since you're interested in science, you decide that it's a perfect place to work. Your boss recently found out that you had expertise in computer programming and has decided that your talents would be put to better use rewriting some software for the science center than feeding the exotic fish.

The science center has some peculiar pricing options, due to its clientele:

Individual Membership - $105/year
Couple Membership - $125/year (any two people)
Family Membership - $145/year (two parents and all of their children)

Adding a nanny to a family membership is an extra $50.
Adding any other additional members to a family membership (limit of two) is an extra $25.

The current software produces a list for each membership order with the number of people in each relevant category for that order. Currently, employees look at the list and manually calculate the cost for each membership. Your job is to write a program to automate the process.

The Problem

Given the number of related adults, number of children in the family, the number of nannies and the number of extra members in an application for membership to the science center, calculate the total cost of the membership.

The Input

The first line of the input file will contain a single positive integer, n ($n \leq 50$), representing the test cases to process. Each of the test cases will follow. Each test case will be on a single line, consisting of four space separated integers: a $(1 \leq a \leq 2)$, the number of related adults for the order, c $(0 \leq c \leq 20)$, the number of children in the family for the order, n $(0 \leq n \leq 5)$, the number of nannies for the order, and e $(0 \leq e \leq 2)$ the number of extra individuals for the order who aren't nannies.

The Output

For each test case, output a single integer on a line by itself representing the cost in dollars for the order.

Sample Input

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

Sample Output

```
105
145
295
```

Orlando City Soccer

Filename: soccer

Soccer has arrived in Orlando! Unfortunately, even the professional soccer club is having trouble with sorting the soccer standings. This is where you come in! Write a program to properly sort all the teams in a soccer league.

Each match results in a win, loss or tie. Each win a team earns is worth 3 points, each tie is worth 1 point and each loss is worth 0 points. Teams are first sorted by total points, from most to least. If two teams have the same number of points, then the team with more wins comes first. Thus, a record of 1-3-0 (W-L-T) is better than a record of 0-1-3. If two teams have the same number of points and the same number of wins, then the next tie breaker is goal differential, the number of goals your team has scored minus the number of goals your opponents have scored, in all of your matches. If this value is tied, then the tie is broken by total number of goals scored.

For the purposes of this problem, you will be guaranteed that no two teams will have the same number of points, wins, goal differential and goals scored. (In MLS, the next tie-breaker is disciplinary points and the list continues on from there, ending in a coin toss or drawing of lots.)

The Problem

Given the score of every game in a soccer league, print out a sorted list of the teams based on the standings from best to worst.

The Input

The first line of the input file will contain a single positive integer, n ($n \leq 50$), representing the number of test cases to process. The test cases will follow. The first test line of each test case will have a single positive integer, k ($2 \leq k \leq 32$), representing the number of teams in the soccer league. The following k lines will have the names of each of the teams, one per line. Each of the names will consist of letters and underscores and will be in between 1 and 50 characters long, inclusive. The next line will have a positive integer, g ($g \leq 5k(k-1)$), representing the number of games played in the league. It is guaranteed that each team will play the same number of games. The following g lines will contain the score for one game each. Each of these lines will have the following four pieces of information separated by spaces, in the order given: home team, goals scored by the home team, away team, goals scored by the away team. The number of goals for each team will always be in between 0 and 100, inclusive and all the team numbers will be valid, and the home and away teams will be two different teams.

The Output

For each test case, output a header line with the following format: `League #X:`

where X is the number of the test case, starting with one.

List each of the teams in order of where they finished in the standings, one team per line.
Follow the output for each case with a blank line.

Sample Input

```
2
2
Orlando_City_Soccer
Seattle_Sounders
1
Orlando_City_Soccer 2 Seattle_Sounders 1
3
a
b
c
3
a 3 b 2
a 2 c 4
b 5 c 1
```

Sample Output

```
League #1:
Orlando_City_Soccer
Seattle_Sounders

League #2:
b
a
c
```

Theme Park Speed

Filename: speed

We can't create a set of problems about activities in Orlando without including a theme park!!!

It's the end of the day and you'd like to beat traffic (if such a thing is even possible). Your goal is to get to your car as fast as possible from your current location. The only problem is that you can only run for a short time. You've mapped out various locations in the park such that some pairs of these locations are connected by undirected paths. For each of these paths, you know how long it takes you to run between the two locations and how long it takes you to walk between the two locations. Because you get winded, if you run on a particular path, you must walk on the following path you take. Given these parameters, write a program to calculate the shortest amount of time it'll take you to get out of the park and to your car!

The Problem

Given a list of locations in a theme park, the walk and run times of the paths connecting locations, and your starting and ending locations, determine the shortest amount of time it will take you to get to your ending location, given that you can't run on two consecutive paths.

The Input

The first line of the input file will contain a single positive integer, n ($n \leq 50$), representing the number of test cases to process. The test cases will follow. The first test line of each test case will have a two positive integer, v ($2 \leq v \leq 100$), and e ($v-1 \leq e \leq 1000$) representing the number of locations in the theme park and number of paths connecting locations, respectively. The second line of each test case will contain two non-negative integers, s ($s < v$) and d ($d < v$), specifying the starting location and destination location, respectively. Note: All of the locations are labeled from 0 to $v-1$. The following e lines will contain the description of each path connecting pairs of locations in the park. Each of these lines will contain the following four space-separated integers: a ($0 \leq a < v$), b ($0 \leq b < v$), w ($2 \leq w \leq 10000$), and r ($1 \leq r < w$), where a and b represent the two distinct locations connected by an undirected path, w represents the number of minutes it takes to walk that path, and r represents the number of minutes it takes to run that path. No two pair of locations will be connected by more than one path and the paths will be such that there will always exist a sequence of connected paths from location s to location d.

The Output

For each test case, output a single positive integer representing the shortest number of minutes you can travel from the designated starting location to the designated destination.

Sample Input

```
2
4 5
0 3
0 1 10 5
0 2 12 5
0 3 20 14
1 3 8 7
2 3 9 3
3 2
0 2
0 1 10 8
1 2 9 6
```

Sample Output

```
13
16
```

Swan Boats

Filename: swan

You've decided to go downtown to the beautiful lake and rent the pedal powered swan boats. Since the rental cost is based on time, you'd like to minimize the time you spend in the boat while visiting each important location in the lake. Write a program to help you calculate the minimum possible cost for your rental. For the purposes of this problem, the lake is a circle centered at (0, 0) with a radius of 1000 meters.

The Problem

Given the starting location of your swan boat, each important location you would like to visit, and your pedaling speed in meters/minute, determine the minimum amount of time necessary to visit each location and return the swan boat to its starting location.

The Input

The first line of the input file will contain a single positive integer, n ($n \leq 50$), representing the test cases to process. Each of the test cases will follow. The first line of each test case will contain three space separated integers: k *(1 ≤ k ≤ 10)*, the number of important locations, a *(0 ≤ a < 360)*, representing the degree measure of the starting point in relation to the center of the lake, and s *(1 ≤ s ≤ 100),* representing your pedaling speed in meters per minute. (0^o represents east, 90^o represents north, 180^o represents west, and 270^o represents south.) The starting point is always on shore, exactly 1000 meters from the center of the lake. The following k lines will contain information about each important location. The i^{th} of these lines will have two non-negative integers, d_i *(1 ≤ d_i ≤ 1000)*, representing the distance of the i^{th} important location from the center of the lake, and a_i *(0 ≤ a_i < 360)*, representing the angle from the center of the lake of the i^{th} important location.

The Output

Output a single positive real number rounded to two decimal places representing the minimum time, in minutes, your journey could take.

Sample Input

```
2
1 0 100
1000 180
5 90 10
80 30
1000 180
280 40
678 235
995 209
```

Sample Output

```
40.00
420.12
```