

1 - Quick Sum
Input File: QuickSumIn.txt

Just before recess, Freddy Gauss was playing with his cell phone while his fellow students were busy practicing their addition skills by adding up the numbers between 1 and 40 inclusive. When his teacher, Ms. Vikki, became aware of this, she told him he could not go to recess until he added up the numbers between 1 and 12,468 inclusive. After a few seconds, Freddy wrote the answer on the board, and was the first to leave for recess.

Faced with the incentive of leaving for recess early, Freddy derived a simple formula to compute the sum, **s**, of the integers from 1 to **n** inclusive: $s = n * (n+1) / 2$. After one addition, one multiplication, and one division he wrote the sum, 77,731,746 on the blackboard and left for recess.

Inputs:

There will be one line of input that contains one integer, the value of **n**.

Outputs:

There will be one line of output that contains the sum of the integers from 1 to **n** inclusive, annotated exactly as shown below.

Sample Input

12345

Resulting Output

The sum of the integers from 1 to 12345 is: 76205685

2 - Baseball Draft

Input File: BaseballDraftIn.txt

In preparation for the spring major league baseball draft, college baseball players are paired based on the number of home runs they have hit while in college. Then, during the draft, each team picks one pair of players to join their team. In order to maintain parity within the league, the player who has hit the least home runs is paired with the player who has hit the most home runs, and this pair of players is added to the list of draftable players. This process is repeated with the remaining players, until all players are paired and placed on the list of draftable players.

For example, if there were 10 players in the draft and their college home run totals were: 20, 15, 30, 3, 40, 28, 16, 50, 21, 36 the player home run parings would be (3,50) (15,40) (16,36) (20,30) (21,28). For this draft, the 5th pairing yields the lowest total home runs, 49, and the 2nd paring yields the highest total home runs, 55.

Given the number of players in the draft and the number of home runs hit by each player while in college, your task is to determine the home run totals for the pairings with the lowest and highest total number of home runs.

Inputs:

The first line contains an integer indicating how many drafts to consider. This is followed two lines of input per draft. The first of these lines contains one integer that represents the number of players in the draft, **n**, which is an even number greater than or equal to 4. The second of these lines contains **n** integers that represent the number of home runs hit by each of the **n** players in the draft while they were in college. Multiple inputs on one line are separated by a space.

Outputs:

For each draft there will be one line of output that contains two integers separated by a space. The first integer will represent the *lowest* total collegiate home runs hit by any of the pairings in the draft. The second integer will represent the *highest* total collegiate home runs hit by any of the pairings in the draft.

Sample Inputs

```
2
10
20 15 30 3 40 28 16 50 21 36
6
10 8 6 4 2 1
```

Resulting Outputs

```
49 55
10 11
```

3 - Largest Number

Input File: LargestNumberIn.txt

Given a string \mathbf{v} that contains only digits and possibly a decimal point, rearrange the digits in it to produce the largest possible number, \mathbf{v}_{\max} . If the string \mathbf{v} contains a decimal point, all of the digits in \mathbf{v} to the right of the decimal point will be to the right of the decimal point in \mathbf{v}_{\max} , and all of the digits in \mathbf{v} to the left of the decimal point will be to the left of the decimal point in \mathbf{v}_{\max} . If \mathbf{v} does not contain a decimal point, \mathbf{v}_{\max} will not contain one.

Inputs:

The first line contains an integer, \mathbf{n} , indicating how many strings are to be converted to their maximum value. This is followed \mathbf{n} lines of input each containing one string, \mathbf{v} , whose digits are to be rearranged to produce the largest possible number, \mathbf{v}_{\max} , as described above.

Outputs:

For each input string, \mathbf{v} , there will be one line of output that contains the digits in \mathbf{v} rearranged, as described above, to produce the largest possible value, \mathbf{v}_{\max} .

Sample Inputs

4
76050659.30054
7605065930054.
.7605065930054
7605065930054

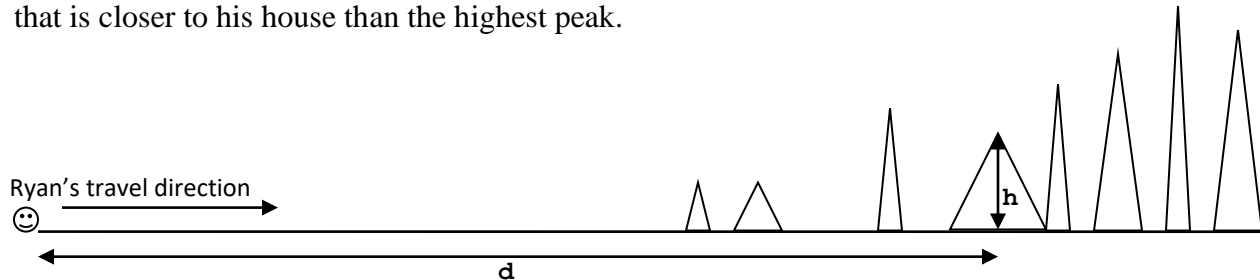
Resulting Outputs

97665500.54300
9766555430000.
.9766555430000
9766555430000

4 - Peak View

Input File: PeakViewIn.txt

Ryan lives in a two dimensional world, and purchased a house from which he can see the highest peak in a mountain range. Each morning he takes run, along a horizontal road, from his house towards the mountains enjoying the view of the highest peak. Having run this route before, he knows that eventually his view of the highest peak will be obstructed by a smaller mountain peak that is closer to his house than the highest peak.



Your task is to determine how far he can run from his house until he can no longer see the highest peak, given the horizontal distance from his house to each peak, d , and the height of each peak above the roadway, h . You should assume the height of his eye above the road is negligible, that distances are measured from his house, and that the units of the distances and peak heights are the same (e.g., kilometers).

Inputs:

The first line contains one integer indicating how many of Ryan's runs to consider. This is followed by one set of inputs per run. The first line in the set contains one integer that represents the number of mountain peaks along Ryan's route. This is followed by one line of input per peak that contains two real numbers (doubles) separated by a space. The first of these numbers represents the horizontal distance from Ryan's house to the peak, d , and the second number represents the height of that peak, h .

Outputs:

For each run there will be one line of output that contains one real number rounded up to two decimal places. This number will be the distance Ryan can run from his house until he can no longer see the highest peak.

(Sample Inputs and Resulting Outputs are given on the next page)

Sample Inputs

3
8
42.0 6.0
34.0 3.0
50.0 10.0
39.0 7.0
47.0 11.0
32.0 3.5
43.0 8.7
45.0 9.9
8
42.0 6.0
34.0 3.0
50.0 12.0
39.0 7.0
47.0 11.0
32.0 3.5
43.0 8.7
45.0 9.9
8
42.0 6.0
34.0 3.0
50.0 10.0
39.0 7.0
47.0 11.0
32.0 5.0
43.0 8.7
45.0 9.9

Resulting Output

25.00
14.00
19.50

5 - Ray the Roofer

Input File: RayTheRooferIn.txt

For the past 30 years Ray has been installing roofs on three different types of houses: Type A, Type B and Type C, each of which requires a different amount of the three types of roofing materials: plywood, shingles, and nails. Now Ray would like to retire and purchase a boat, which he would store in his garage. However, his garage is filled with left over packages of roofing material. To remedy this situation he has decided to work one more year, so that he can use up *all* of the packages of materials in his garage.

Hearing about Ray's return to roofing, a builder of housing developments has asked him to roof the houses in a development planned for next year, which will include a mix of the three types of houses. Ray agreed to roof the houses under the condition that Ray could specify the *number* of *each* of the three types of houses that would be constructed, so that he could use up *all* of the packages of materials in his garage. The builder agreed to this condition. Knowing the packages (AKA units) of the materials required for each house type, and the materials in his garage, Ray composed the below table.

	Packages (AKA units) of		
	Plywood	Shingles	Nails
House type A requires	3	5	29
House type B requires	7	11	31
House type C requires	19	23	37
Ray's material inventory	206	270	598

After analyzing the table for 3 long days, Ray concluded that he would use up all of his roofing materials if the builder built 4 type A houses, 6 type B houses, and 8 type C houses, because:

$206 \text{ units of plywood} = 3 \text{ units} * 4 \text{ type A houses} + 7 \text{ units} * 6 \text{ type B houses} + 19 \text{ units} * 8 \text{ type C houses}$
 $270 \text{ units of shingles} = 5 \text{ units} * 4 \text{ type A houses} + 11 \text{ units} * 6 \text{ type B houses} + 23 \text{ units} * 8 \text{ type C houses}$
 $598 \text{ units of nails} = 29 \text{ units} * 4 \text{ type A houses} + 31 \text{ units} * 6 \text{ type B houses} + 37 \text{ units} * 8 \text{ type C houses}$

Your task is to automate Ray's analysis for different versions of the above table.

Inputs:

The first line contains an integer, **n**, indicating how many tables to consider. This is followed by four more lines per table, containing three integers each that are the table's data. The three integers on lines 1, 2, and 3 represent the number of plywood, shingles, and nails packages required for house type A (line 1), B (line 2) and C (line 3) respectively. The three integers on line 4 represent the number of plywood, shingles, and nails units in Ray's garage. Multiple inputs on a line are separated by a space.

Outputs:

For each of the **n** tables considered, there should be one line of output containing three integers each separated by a space. The first of these is the number of Type A houses, the second the number of Type B houses, and the third the number of Type C houses that when constructed use all of the materials in Ray's garage. If there is no combination of Type A, B, and C houses that can be constructed using all of the materials in Ray's garage, the output will be three zeros.

(Sample Inputs and Resultant Outputs are on the next page)

Sample Inputs

3
3 5 29
7 11 31
19 23 37
206 270 598
3 5 29
7 11 31
19 23 37
71 99 283
9 16 29
7 11 31
19 23 37
183 99 270

Resultant Outputs

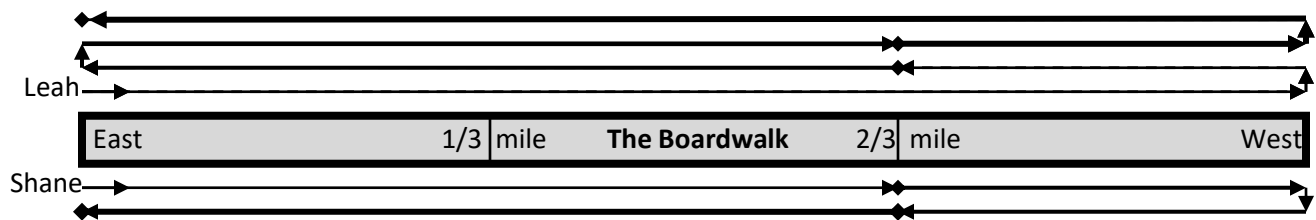
4 6 8
4 3 2
0 0 0

6 - Runners

Input File: RunnersIn.txt

Shane and his friend Leah travel to the Coney Island Beach every day to run laps on the boardwalk, which is aligned east-west. They begin each lap at the east end of the boardwalk, run to the west end, and then turn around and run back to its east end. Although they both run at a constant pace, Leah always runs faster than Shane, and so they only can converse when they pass each other.

If the boardwalk is 1 mile long and Leah ran at a pace of 7 minutes per mile, and Shane's pace was 14 minutes per mile, their first two meetings would be at the $2/3$ mile marker, and their third meeting would be at the east end of the boardwalk as shown below (♦ indicates a meeting).



Your task is to determine the location of the 1st, 2nd, 3rd, ... n^{th} meetings as they run their daily laps, given the length of the boardwalk, and their running speeds. You should assume that the U turns they make, at the beginning and end of the boardwalk, do not consume any time.

Inputs:

The first line contains a positive integer indicating how many runs to consider. This will be followed by one line per run that contains three real numbers, followed by an integer. The first of these represents the length of the boardwalk in miles, the second and third numbers represent Shane's and Leah's running paces in minutes per mile, and fourth number is the number of meetings, n , during the run whose location is to be determined. All inputs on a line will be separated by a space.

Outputs:

There will be one output per line per run that contains n real numbers, each separated by space, formatted with a leading zero and two digits of precision rounded up. The first of these will represent the distance from the east end of the boardwalk to the point of the 1st meeting, the second the distance from the east end of the boardwalk to the 2nd meeting, ..., and the last the distance from the east end of the boardwalk to the n^{th} meeting.

Sample Inputs

```
3
1.0 14.0 7.0 6
2.0 15.0 13.5 6
1.0 15.0 5.0 6
```

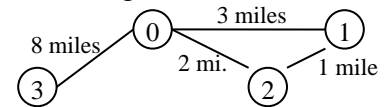
Resultant Output

```
0.67 0.67 0.00 0.67 0.67 0.00
1.89 0.21 1.68 0.42 1.47 0.63
0.50 1.00 0.50 0.00 0.50 1.00
```


7 - Plowing

Input File: PlowingIn.txt

Park Ranger Jane's staff maintains the hiking trails that begin and end at **n** observation points within the park. From any observation point, a hiker can take one, or more, bi-directional trails that lead directly to another observation point. Each observation point has been assigned a number from 0 to **n**-1, as show in the below figure. Every observation point can be reached from any other observation point. The trails are arranged such that the hike between two points may be a direct route (e.g., 3 to 0), or it may be an indirect route (e.g., 2 to 0 to 3) or both (e.g., 0 to 1, and 0 to 2 to 1).



In order to open the park as quickly as possible after a snow storm, Jane would like to give her staff a list of a subset of trails to plow such that every observation point is accessible, and the total distance to be plowed is a minimum. Your task is to produce the list of trails to plow.

Inputs

The first line of input contains the number of parks to consider, followed by one set of inputs per park. The first line in the each set contains an integer that specifies the number of observation points in that park, **n**, followed by one line of integers per observation point in the park. The first input on each of these lines will be an observation point's number, followed by the number of trails (**t**) leaving that point, followed by **t** pairs of integers. The first integer in each pair will be the observation point at the other end of the trail, and the second integer in the pair will be the length of the trail in miles. All inputs will be separated by a space.

Outputs

There will be one group of outputs per park considered. The first line of output in the group will contain two integers that represent the minimum number of miles of trails to be plowed such that every observation point is accessible, followed by the total miles of trails in the park. This will be followed by one line of output per trail to be plowed, which contains the two observation point numbers that the trail directly connects in ascending order. These observation point pairs will be output in ascending order based on the first point number in the pair, as shown in the sample output below. Each output on a line separated by one space.

Sample Inputs

```

2
4
0 3 1 4 2 2 3 8
1 2 0 3 2 1
2 2 0 2 1 1
3 1 0 8
7
0 2 2 3 4 6
1 3 2 4 4 1 6 10
2 3 0 3 1 4 4 6
3 1 5 15
4 4 0 6 1 1 2 6 5 9
5 2 3 15 4 9
6 1 1 10
  
```

Resulting Outputs

```

11 14
0 2
0 3
1 2
42 54
0 2
1 2
1 4
1 6
3 5
4 5
  
```