acm International Collegiate Programming Contest 2014 IBM event sponsor

South Pacific Region

Eastern Divisional Contest

30th August 2014

Problem Set

Problem A **Ages** **Time Limit: 1s**

According to Guinness World Records,

- The youngest player to score a competitive maximum break in snooker was Ronnie O'Sullivan (UK, b. 5 December 1975), who was 15 years 98 days old when he achieved the feat during the English Amateur Championship (Southern Area) at Aldershot, Hampshire, UK, on 13 March 1991.
- Princess Alice, Duchess of Gloucester (UK, b. Lady Alice Christabel Montagu Douglas Scott, 25 December 1901), … died peacefully in her sleep at the age of 102 years 309[1] days on 29 October 2004 at Kensington Palace, London, UK.

This problem requires you to work out the age of a person in years and days when a particular event occurred, and present that age along with the name of the person and the event being described.

Input

The first line of input will be a single positive integer, N, being the number of events presented. Each event will be presented as 4 lines of input as follows:

Line 1 : the name of the person involved. There will be no more than 40 characters.

Line 2 : a description of the event. There will be no more than 60 characters.

Line 3 : the date of birth of the person in the format yyyy mm dd.

Line 4 : the date the event occurred in the format yyyy mm dd.

The dates on lines 3 and 4 are legal dates from the Gregorian calendar. The date of birth will not come after the date of the event.

Output

For each event, one line of output is required. The format is:

`<name> was Y years and D days old when <event>.`

`<name>` is the name of the person as defined in line 1 of the input, followed by a space.

`Y` and `D` are the age of the person in years and days respectively at the time of the event.

`<event>` is the description of the event as defined in line 2 of the input. It is preceded by a space and followed by a full stop.

1 The figure of 309 days had to be corrected from 308 days when they remembered that 2004 was a leap year, so February 29th had to be included.

Note

In most cases, a year is a leap year if it can be divided exactly by 4, so 2008 and 2012 were leap years, 2009 and 2013 were not. When the year can be divided exactly by 100, however, it is not a leap year unless it can be divided exactly by 400, so 1800 and 1900 were not leap years but 2000 was.

Sample Input

```
2

Ronnie O'Sullivan

he scored a competitive maximum break in snooker

1975 12 05

1991 03 13

Princess Alice, Duchess of Gloucester

she died peacefully in her sleep

1901 12 25

2004 10 29
```

Output for Sample Input.

```
Ronnie O'Sullivan was 15 years and 98 days old when he scored a competitive maximum break in snooker.

Princess Alice, Duchess of Gloucester was 102 years and 309 days old when she died peacefully in her sleep.
```

Problem B **Sequences** **Time limit: 2s**

There is a well known sequence that goes:

`1, 11, 21, 1211, 111221, …`

where each successive term describes the previous term using run length encoding. That is, the first term (1) is described by "one one" (a sequence of length one consisting entirely of ones). Thus the next term is 11. This, likewise may be describe as two ones (a sequence of length two consisting entirely of ones), and so on. Your task is to write a program that given n returns the nth term of this sequence (where 1 is the first term and 11 is the second term).

However, rather than always starting with 1, we will allow the sequences to start with any number less than 10000. So for example the sequence starting at 333 is:

`333, 33, 23, 1213, 11121113, 31123113` etc.

Input

The input will start with an integer N (N<100), the number of cases.

The following N lines each consist of 2 integers, S and T. S is the starting number (`0 < S < 10000`) and T is the term required (`0 < T < 30`).

Output

For each case print the Tth term of the sequence starting at the given number S.

Sample Input:

```
2

1 4

333 5
```

Sample Output

```
1211

11121113
```

Problem C **Falling Stock Prices** **Limit: 10s**

Joe has bought some stock of a company called Automatic Computing Machines (ACM) and is now getting nervous about his decision due to falling prices over time. He has decided to sell his stock whenever there is an overall downward trend from the initial price over a significant time period.

You are given a sequence of stock prices P = p_1, p_2, p_3, ... , p_n for the last n days for company ACM. Your task is to help Joe find the largest length k of falling trend (longest subsequence) S = s_1, s_2, s_3, ... , s_k of P, where $s_1 = p_1$ is included (day stock was bought) and $s_i > s_{i+1}$ for $1 <= i < k <= n$. If k is too big then we want to urge Joe to sell his stock which he bought on day 1 at price p_1.

Input

The input consists of a series of up to 200 test cases, one per line. Each line has up to 2000 positive integers denoting the daily stock prices, in order, starting at day one. The final line contains just 0 – do not process this line.

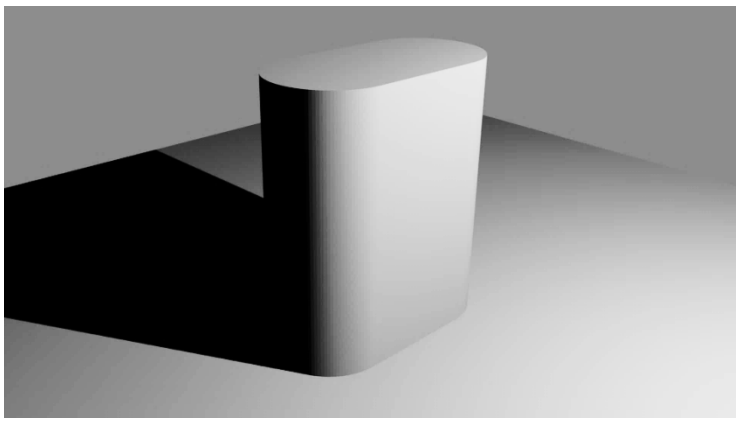
Output

For each test case output a single line showing the maximum falling stock trend length.

Sample Input	**Output**	**Explanation**
`20 10 10 5 8`	`3`	The price sequence 20,10,10,8 is not a falling trend for the 1st input since we need strictly decreasing prices. Thus 20,10,5 and 20,10,8 are the only maximal falling trends of length 3 (hence the maximum is 3).
`10 23 20 3 5 7 2 1 5 3 1`	`5`	The longer decreasing sequence 23,20,7,5,3,1 of length 6 is not considered a trend since it does not start with the initial day's stock price. 10, 7, 5, 3, 1 is the longest sequence.
`0`		

Problem D — Large Headquarters Collider (LHC) — Time Limit: 1s

As you can see from the photograph the ICPC headquarters building is a widened cylindrical concrete tower set in the middle of a large flat concrete plane (entrance is from the underground car park and obviously windows were not needed). Unfortunately, due to a fire in the local planning office and some issues to do with a hacked digital storage facility, there are no longer any plans available for the building. As a start towards reconstructing the plans, ICPC wants to measure the location and size of the base. Strict security protocols require that no-one may approach the walls, so direct measurement is not possible. Keeping well clear of the building, you choose an origin and establish a coordinate system, with x axis aligned to the flat faces of the building. From well back you fire large hard rubber balls at the building and measure the way in which they bounce. The only thing needed now to get the required building dimensions is software. You must write a program. (To allow thorough testing, your program is expected to solve several problems, with a range of sizes of buildings and balls.)



Input

The first input line holds a single integer P – the number of problems to be solved. This is followed by data for each of the problems. Each problem data set starts with a single number N (0 <= N <= 3), being the number of observations, followed by a line for each observation. Observation lines each hold 9 real numbers: s, a_x, a_y, b_x, b_y, c_x, c_y, d_x, d_y. s is the radius of the ball; (a_x, a_y) is the point from which the ball is thrown, (b_x, b_y) is a point soon after the throw but before a hit (if any) on the building. (c_x, c_y) and (d_x, c_y) are successive points as the ball moves away from the building (after any hit). Note: ball positions are for the centre of the ball. All 9 values are in the range 0 to 1000.

Output

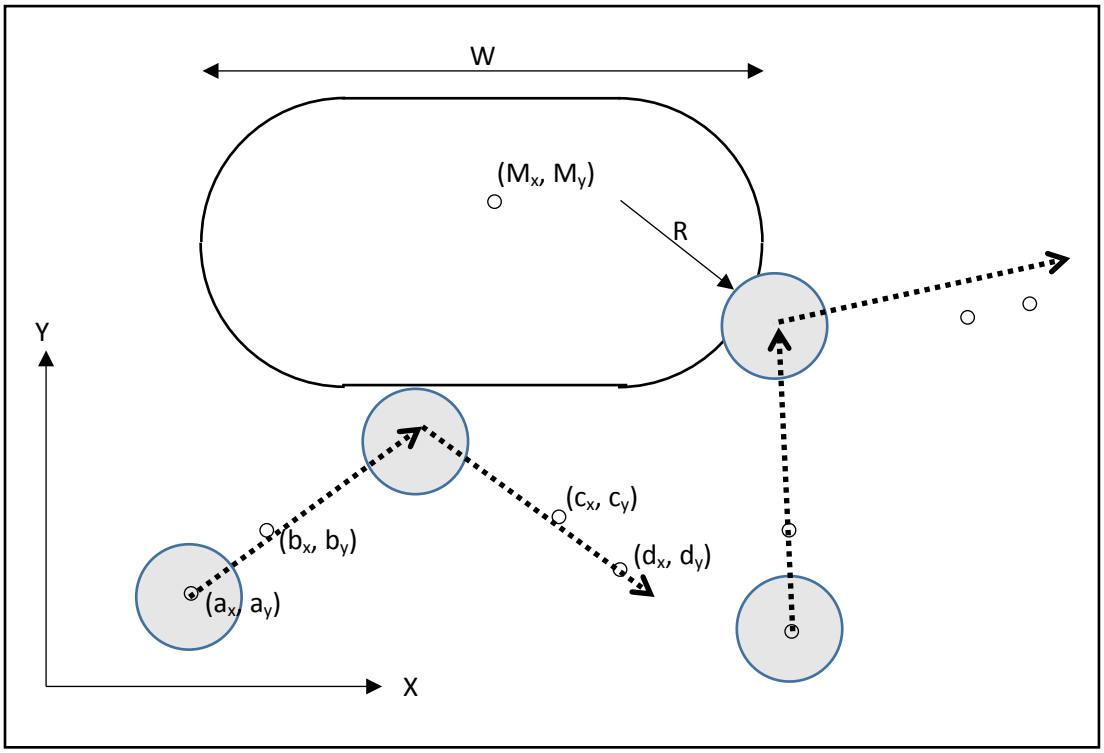
The building base is an axis aligned rectangle with semicircular ends. As shown in the diagram, the location and size are described by four values M_x, M_y, W and R. M_x and M_y are the coordinates of the centre, W is the total width and R is the radius for the ends. Your output for each problem should be a line with those of the four real values that you can determine, each rounded to one decimal place. For values which you cannot determine output 'unknown'.

See diagram on next page.

Notes

- All ball throws are forward at some angle. Ie: $a_y < b_y$
- The balls do not compress significantly on impact.
- Ball bounces are perfect – angle of incidence equals angle of reflection.
- No ball hits the wall straight on (ie: doesn't bounce back on its incoming path).
- All throws hit the building.
- All the throws in a set hit the building at usefully different y coordinates.
- All throws hit the front of the building (impact y coord < m_y.)
- Points a and b, and points c and d for a throw are always usefully far apart.

Where by 'useful' it is meant that reasonable calculations can be used – major errors will not arise as a result of two bounces being too similar to yield different information about the wall – or starting/ending pairs being too close together to give reliable direction vectors, etc.



Sample Input

```
2
0
2
10 620.7107 100 620.7107 200 800 329.2893 900 329.2893
10 300 100 400 200 600 200 700 100
```

Sample Output

```
unknown unknown unknown unknown
unknown 400.0 unknown 90.0
```

Problem E **Chaos** **Time Limit: 2s**

Professor Chaos is about to strike terror into the hearts of the students at South Shore Elementary in revenge against the students at the school who have tormented him. The school dance is a huge annual tradition where the school students partner up in a dance off competition. The dance consists of N boys and N girls who are paired together to compete in the dance off. The boys and girls of the school collectively decide on dance partners before the dance, and submit the pairings to the principal. Using his superior intellect Professor Chaos has gained access to the partners for the upcoming school dance. In an evil plan to unleash social anarchy upon his tormentors, he has decided to change the pairings so that none of the students remain assigned to the partner they were paired with. For example, if Kyle, Stan and Kenny were originally paired with Rebecca, Wendy and Kelly respectively, one configuration which would please Professor Chaos would be if Kyle, Stan and Kenny were paired with Wendy, Kelly and Rebecca respectively.

Professor Chaos has recruited you, General Disarray, as one of his minions. It is up to you to count the number of dance partner configurations that pair the boys with the girls so that none of the pairings remain the same as those that were submitted by the students. Then he can make sure that everyone in the school gets served.

Input

The input starts with an integer T, (1 <= T <= 100) representing the number of test cases on a line by itself. Each of the following T lines contains one integer N, (1 <= N <= 10,000,000), which is the number of pairings, N, that the dance will have.

Output

For each test case the output consists of a single integer on a new line, which is the number of ways that Professor Chaos can arrange the boys and girls so that none of the original pairings remain. You should return your answer modulus 100,000,007.

Sample Input

```
3
2
3
6
```

Output for Sample Input

```
1
2
265
```

Problem F **Statement Ordering** **Time Limit: 30 s**

In a compiler the code generator usually has an 'optimiser' which may reorganise generated instructions to avoid repeated calculations and to make the best use of resources, like pipelines and registers. You are invited to join a group working on optimising register allocation for non-branching sequences of instructions. Their first idea is to solve the problem by brute force – simply to try all permissible orders for instructions and choose the best one. That sounds easy, but the problem is in the phrase "all permissible orders" – it really means "all orders which give the results that the programmer specified". For example, consider the statements

```
A = B + C
D = A
```

If we swapped the order of execution of the lines, the value stored in D at the end might be quite different from that intended. Your task is to write a program which counts all "permissible" orderings for a sequence of statements.

In the compiler context in which you are working, preliminary code generation will have already been completed. The program being compiled has been converted into a sequence of simple instructions: either moving a value or doing a single arithmetic operation. For example a statement like

```
A = B * (C + D) + E * (F + G)
```

would have been converted to

```
r0 = C + D
r1 = B * r0
r2 = F + G
r3 = E * r2
r4 = r1 + r3
A = r4
```

Extra 'variables' r0, r1, etc. have been introduced to hold intermediate values. The compiler group is working towards deciding how to allocate registers to hold those (where possible). The values occurring in instructions can be variables (Upper case single letters: A, B, C, etc) or temporary storage locations (written as r1, r2, r3, …)). Only operators + and * will occur.

Input

Input consists of a sequence of problems. The input for each problem starts with a line holding a single integer N, being the number of statements for that problem. It is followed by N lines holding statements. Statements take the form

```
<var> = <var>    or    <var> = <var> <op> <var>
```

Where <var> can be an uppercase letter of the alphabet, or a lowercase 'r' with a number. <op> can be '+' or '*'. The tokens on each line are separated by single spaces. 1 <= N <= 55. End of input is by a value 0 for N. Note that the number of possible orderings of 55 could be extremely large. Your program will only be given data for which there are many dependencies between the statements, ensuring that the number of permissible orderings will never exceed 2 million.

Output

A single line of output is required for each problem. The line should hold an integer, being the number of “permissible” orderings of the given sequence of statements. The requirements for a permissible ordering are:

- All statements must be included. It is NOT your task to identify or remove useless statements.
- If a variable is defined in the sequence, that definition must precede uses.
- If a variable is used before being defined, those uses must precede the definition. Ie: Your statements are part of a bigger program and variables will have been defined earlier.
- You may assume that no variable is defined more than once in the sequence.

Sample Input

```
4
r1 = A + B
r2 = C + D
E = r1 + r2
F = G
6
r1 = A + E
r2 = C + D
E = r1 + r2
F = E
G = E
H = E
0
```

Output For Sample Input

```
8
12
```

Problem G **Time Zones** **Time Limit: 3s**

The organisers of a global programming contest have interest from university teams from all parts of the world who want to take part in their regular practice contests. The teams need to be at their university computer laboratory to participate in these contests and each university's computer laboratory has different opening and closing hours. For a team to participate in a contest their university computer laboratory must be open for the entire duration of the contest.

Given the starting time and duration of a contest and the opening hours of the universities' computer laboratories work out which universities can take part in a contest.

Input

The first line of input contains a single integer T (T <= 100) being the number of test cases. For each test case the first line of input contains a single integer U (1 <= U <=1000), being the number of teams who are interested in participating in the contests.

The following U lines contain the team names followed by a colon, the UTC (Coordinated Universal Time) time zone in which the university is situated and the opening and closing times for the university's computer laboratory. No team name will contain a colon. Some computer laboratories may be open 24 hours a day; in these cases, the opening and closing times will be the same.

The final line of input for each test case contains the time zone of the contest organiser followed by the starting time of the contest and the duration, D, of the contest in minutes 30 <= D <= 600.

All times are in hh:mm am/pm format in the local time zone (01 <= hh <= 12 and 00 mm <= 59). Note that 12:00 am is midnight or 00:00 in 24 hour clock format.

All time zones are in the format UTC+/-HH being the number of hours ahead or behind UTC (-12 <= HH <= +14). UTC+00 is the time zone used as the basis of Coordinated Universal Time and all time zones are whole numbers of hours.

Output

For each test case output the number of teams that can participate in the contest on a single line followed by the names of the teams which can participate in the contest in alphabetical order. Separate the output of each test case with a blank line

Sample Input

```
2
3
Hackers: UTC+10 09:00 am 09:00 pm
Crackers:UTC-08 01:00 am 11:00 pm Cracked Actors: UTC+00 08:00 am 08:00 pm UTC+10 12:00 pm 300
3
University of ABC: UTC-03 05:00 am 03:00 am
Crackers: UTC-06 01:00 am 11:00 pm
Team GHI: UTC+00 12:00 am 08:00 pm
UTC-04 09:00 am 300
1197.50
```

Output For Sample Input

```
1
Hackers
3
Crackers
Team GHI
University of ABC
```

Problem H **Trivial** **Time Limit: 3s**

A group of friends like to get together each week to take part in trivia contests that seem to be on offer in just about every pub and club around the city. They live in different parts of the city and would like to work out which one trivia contest they should go to each week. They each have travel costs to and from the trivia venue and meal costs each week but can offset those costs by winning weekly and/or jackpot prizes at the trivia contest.

They have done some research at the various trivia contests around the city. They have mapped out possible bi-directional road segments that lead to the different trivia venues from their homes. Each person has a different type of car and knows the travel costs for their car per kilometre travelled. The cost of meals per person at each venue is known and they have also worked out the probability of winning the weekly prize and the jackpot prize at each venue.

Your task is to write a program to recommend to which trivia contest the friends should go each week to minimise their overall costs or maximise their winnings.

Input

Each test case begins with four integers P, V, R and L on a line by themselves. P ($1 \le P \le 10$) is the number of people in the group of friends. V ($1 \le V \le 10$) is the number of trivia venues the friends have researched. R ($1 \le R \le 500$) is the number of road segments the friends could use to travel between their homes and the trivia venues. L ($2 \le L \le 250$) is the number of locations in the test case. Locations can be a person's house, a trivia venue or the end of a road segment. Houses and trivia venues are situated at the end of a road segment and a house will not have the same location number as a trivia venue. There will be up to 100 test cases.

The next P lines contain information about each person in the group. Each line begins with an integer being the location number L_i ($1 \le L_i \le L$) where person P_i lives. This is followed by a floating point number, T_i ($0.0 \le T_i \le 10.0$), the travel cost per kilometre for the person.

The next V lines contain information about each trivia venue. Each line begins with an integer being the location number L_j ($1 \le L_j \le L$) of the trivia venue. This is followed by three integers - M_j, W_j and J_j - and two floating point numbers - A_j and B_j. M_j ($0 \le M_j \le 50$) is the average meal cost per person for venue V_j. W_j ($0 \le W_j \le 100$) is the weekly prize amount for winning the trivia contest at venue V_j. J_j ($0 \le J_j \le 1\,000$) is the prize for answering the jackpot questions correctly at venue V_j. A_j and B_j ($0.0 \le A_j, B_j \le 1.0$) are the probabilities of winning the weekly prize and the jackpot prize respectively.

The next R lines contain information about the road segments. Each line consists of three integers U_k, V_k and D_k. U_k and V_k ($1 \le U_k, V_k \le L$) are location numbers being the end points of road segment R_k. D_k ($1 \le D_k \le 100$) is the length of the road segment in kilometres. Each road segment is bi-directional.

Input ends when P, V, R and L are all equal to 0.

Output

For each test case, output on a line by itself the venue number preceded by the word "Venue" which minimises the costs for the group of friends attending a weekly trivia contest. If the total weekly costs are identical for two or more venues output the venue which minimises the total travel distance for the group as they are concerned about their amount of travel on the environment. If there is more than one venue which minimises the total travel distance, minimise the venue number.

Sample Input

```
2 2 4 5
1 0.15
2 0.25
3 20 10 100 0.5 0.1
4 10 20 500 0.8 0.05
1 5 20
2 5 10
3 5 10
4 5 20
0 0 0 0
```

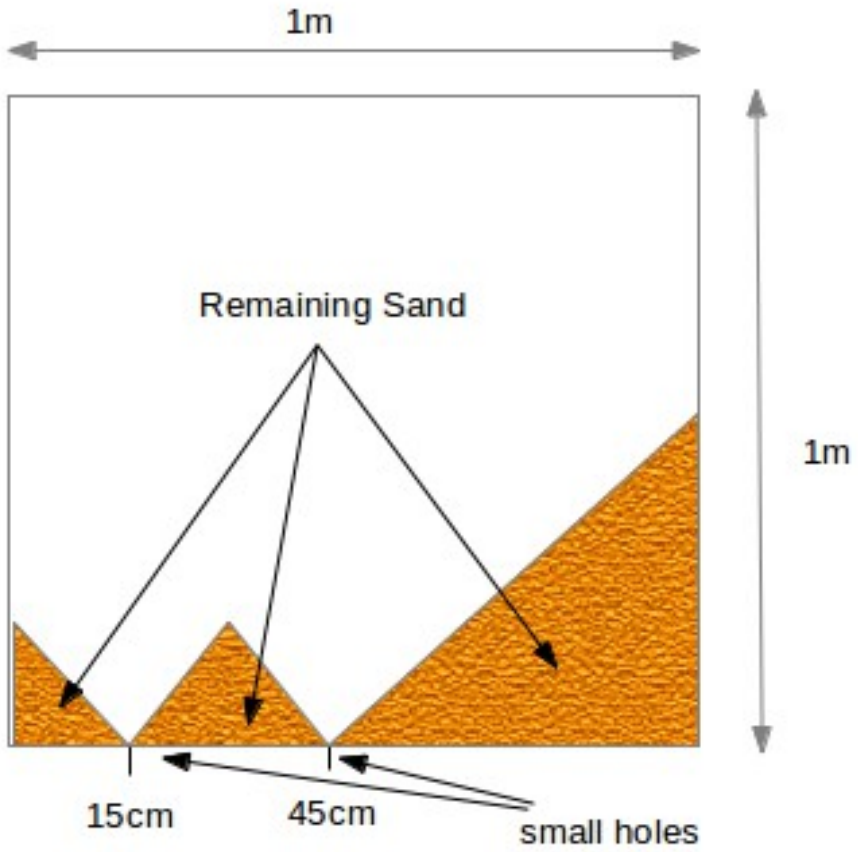
Output for Sample Input

```
Venue 2
```

Problem I **Sand Drawings** **Time Limit: 1s**

An artist wishes to make an artwork that has sand trapped between two plates of glass. The glass plates are in the same plane and near to each other, they also have thin side walls and a thin bottom to trap the sand. The artwork is exactly 1 metre wide and 1 metre high and is initially completely full of sand.

To make the artwork more interesting the artist makes small holes in the bottom of the artwork letting sand escape. The sand will continue to escape until there are 45 degree walls on either side of the small holes. Given a number of small holes your task is to calculate the area of the remaining sand (Note: once this is done the artist would block the holes and add extra sand of a complementary colour to stabilise and complete the artwork but this is not your concern).



The diagram is of an artwork after small holes at 15 cm and 45 cm have been placed along the bottom. The leftmost triangle of this artwork has an area of 112.5 cm^2, the middle has an area of 225 cm^2, and the rightmost triangle has an area of 1512.5 cm^2. So the total area of the sand for this artwork is 1850 cm^2.

Input

The input consists of a number of test cases T (T <= 200), with one test case per line. Each test case starts with an integer N (0 <= N <= 101), the number of holes for the test case. The following N unique unordered integers, H_i (0 <= H_i <= 100), describe the placement of the holes in cm along the bottom of the artwork from the left edge. A hole is specified once only. The test cases end with a value for N of -1.

Output

For each test case output one line being the area in cm^2 of sand after the hole(s) have been placed. Display the output as a floating point number with exactly 2 decimal places.

Sample Input

```
1 50
2 15 25
3 3 42 99
-1
```

Output for Sample Input

```
2500.00
2950.00
1197.50
```