2019 FIU High School Programming Competition

Division 1 Problems

A	Firefly
B	Runes
C	Word Ladder
D	Wood Cutting
E	Headset Sales
F	Troop Mobilization
G	Party Animals
H	Miles to Kilometers
I	Bishop Checking
J	Interpreter

A – Firefly

Problem Statement

A Japanese firefly has flown into a cave full of obstacles: stalagmites (rising from the floor) and stalactites (hanging from the ceiling). The cave is N units long (where N is even) and H units high. The first obstacle is a stalagmite after which stalactites and stalagmites alternate. Here is an example cave 14 units long and 5 units high:

The Japanese firefly is not the type that would fly around the obstacle; instead it chooses a single height and rams itself from one end of the cave to the other, destroying all obstacles in its path with its mighty ninja moves. In the example shown here, choosing the 4^{th} level up from the ground forces the firefly to destroy eight obstacles. This is not the best choice, because the firefly will end up less tired if it chooses either level one or five, as they would require destroying only seven obstacles.

Your task is to write a program that determines the minimum number of obstacles the firefly needs to destroy to reach the end of the cave, and on how many distinct levels it can achieve that number.

Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. The first input line contains two integers N and H ($2 \leq N \leq 200000$ and $2 \leq H \leq 500000$), the length and height of the cave. N will always be even. The next N lines contain the sizes of the obstacles, one per line. All sizes are positive integers less than H.

Output Description

Output two integers separated by a single space on a single line. The first number is the minimum number of obstacles the firefly has to destroy and the second is the number of levels on which that can be achieved.

The following sample input corresponds to the diagram shown in the description.

Sample Input 1	Sample Output 1
14 5 1 3 4 2 2 4 3 4 3 3 3 2 3 3	7 2

Source: Croatian Regional Competition in Informatics 2007

B - Runes

Problem Statement

You are helping an archaeologist decipher some runes. She knows this ancient society used a Base 10 system, and they never start a number with a leading zero unless it is exactly zero (single digit). She's figured out most of the digits, as well as a few operators, but she needs your help to figure out the rest.

The professor will give you a simple math expression. She converted all of the runes she knows into digits. The only operators she knows are addition (+), subtraction (−), and multiplication (*), so those are the only ones that will appear. Each number will be in the range from −999999 to 999999, and will consist of only the digits 0-9, possibly a leading −, and possibly a few **?**s. The **?**s represent a digit rune that the professor doesn't know (never an operator, an =, or a leading −). All the **?**s in an expression will represent the same digit (0-9), and it won't be one of the other given digits in the expression.

Given an expression, figure out the value of the rune represented by the question mark. If more than one digit works, give the lowest one. If no digit works, well, that's bad news for the professor -- it means that she's gotten some of her runes wrong. Output −1 in that case.

Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. Each test case will consist of a single line, of the form:

[number][op][number]=[number]

Each *[number]* will consist of only the digits 0-9, with possibly a single leading minus (−), and possibly some ?s. No number will begin with a leading 0 unless it is 0, no number will begin with −0, and no number will have more than 6 places (digits or ?s). The *[op]* will separate the first and second *[number]*s, and will be one of: +, − or *. The **=** will always be present between the second and third *[number]*s. There will be no spaces, tabs, or other characters. There is guaranteed to be at least one ? in every equation.

Output

Output a single line with the lowest digit that will make the equation work when substituted for the **?**s, or output −1 if no digit will work. Output no spaces.

Sample Inputs	Sample Outputs
1+1=?	2
123*45?=5?088	6
-5?*-1=5?	0
19--45=5?	-1
??*??=302?	5

Source: 2014 ICPC Southeast USA Regional Contest

C – Word Ladder

Problem Statement

A Word Ladder is a puzzle in which you transform one word into another word of the same length by changing one letter at a time. But, there's a catch: every word that you form in each step must be in the dictionary! Here's an example of how to transform CAT into GAS:

CAT → CAR → WAR → WAS → GAS

Of course, you want to use the fewest number of transitions possible. These puzzles can be tough, and often you'll think to yourself: "Darn it! If only [some word] was in the dictionary!"

Well, now is your chance! Given a dictionary, and a starting and ending word, what ONE single word could you add to the dictionary to minimize the number of steps to get from the starting word to the ending word, changing only one letter at a time, and making sure that every word at every step is in the dictionary?

Input Description

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will start with a line with a single integer n ($2 \leq n \leq 1000$) which indicates the number of words in the dictionary. The dictionary will follow on the next n lines, with one word per line. All words will consist of between 1 and 8 capital letters only, and all of the words in a test case will be of the same length. The first word in the list will be the starting word of the word ladder, and the second will be the ending word of the word ladder.

Output Description

Output exactly two lines. The first line holds the one single word that you would add to the dictionary, and the second holds an integer indicating the minimum number of steps to get from the starting word to the ending word after adding your word.

- It is possible that there is more than one word you can add that will make your path as short as possible. In this case, output the solution word that comes first alphabetically.
- It is also possible that there is no word you can add that will that will make your path any shorter. In this case, output 0 (zero) as the word.
- It is possible that there's no word you can add that makes the solution possible. In this case, output 0 (zero) as the word, and -1 as the number of steps.

Sample Input 1	Sample Output 1
3 CAT DOG COT	COG 3

Sample Input 2	Sample Output 2
2 CAT DOG	0 -1

Source: 2014 Southeast USA Regionals Division 1

Sample Input 3	Sample Output 3
4 CAT DOG COT COG	0 3

Source: 2014 Southeast USA Regionals Division 1

D – Wood Cutting

Problem Statement

Maria recently decided to get into the wood cutting business, and now has N customers who would like to have their wood cut. A piece of wood of size s takes s time units to cut, and she can only cut a single piece of wood at a time. To maximize customer happiness, she would like to minimize the average total time a customer has to wait to have all of their wood pieces cut. Can you help her?

Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. Each test case begins with an integer N ($1 \leq N \leq 500$), the number of customers. N lines follow, where the ith line begins with an integer W_i ($1 \leq W_i \leq 1000$), the number of wood pieces that customer i would like to have cut, followed by W_i integers, the sizes of the wood pieces owned by the customer. Each wood piece has a size between 1 and 1000 inclusive.

Output Description

For each test case, output a single line containing the minimum average total time a customer will wait to have all their wood pieces cut. Your answer must be rounded to the first 7 digits after the decimal point.

Sample Input 1	Sample Output 1
1 2 7 5	12.0000000

Sample Input 2	Sample Output 2
2 1 7 1 5	8.5000000

E – Headset Sales

Problem Statement

Carina plans to sail from Miami to Atlantis, which is at the diagonally opposite end of the Bermuda Rectangle. To facilitate her planning, Carina has drawn herself a grid and placed Miami at coordinates (0, 0) and Atlantis at coordinates (A_x, A_y) in the northeast direction. Along the way, Carina could stop at various islands where she could sell the VR headsets that she produces in Miami. For each island, she knows how many headsets she can sell. However, the winds will only allow her to sail in the northeasterly direction. This means that if she decides to sail from a point at coordinates (x_1, y_1) to a point at coordinates (x_2, y_2), then $x_1 < x_2$ and $y_1 < y_2$. Help Carina to maximize the sale of VR headsets on her sail from Miami to Atlantis. The input to the problem includes the coordinates of Atlantis and the coordinates of the N islands, along with the number of headsets that she can sell at each of those islands.

Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. The first line of the input will contain three integers, A_x, A_y, and N ($0 \leq A_x, A_y, N \leq 3000$). The first two integers give the coordinates of Atlantis, and N is the number of islands. Each of the following N lines will contain the x, y coordinates and number of headsets she can sell on a single island.

Output

Output a single integer indicating the maximum number of headsets Carina can sell, using all the available islands.

Sample Input 1	Sample Output 1
10 10 3 5 4 500 5 6 600 7 6 500	1000

Sample Input 2	Sample Output 2
10 10 3 5 4 500 5 6 1100 7 6 500	1100

Sample Input 3	Sample Output 3
10 10 3 5 5 500 6 6 1100 7 7 500	2100

F – Troop Mobilization

Problem Statement

You are playing a strategy games in which you are required to mobilize an army. The army consists of different types of troops, each of which has a cost, health, and potency. You can acquire any combination of the troop types, even fractional, such that the total cost is no more than the amount of money you have to spend. The strength of the army is equal to its total health value multiplied by its total potency (i.e. $(\Sigma a_i h_i) \times (\Sigma a_i p_i)$, where $\boldsymbol{h_i}$ is the health of troop type $\boldsymbol{i}$, $\boldsymbol{p_i}$ is the potency of troop type $\boldsymbol{i}$, and $\boldsymbol{a_i}$ is the amount acquired of troop type $\boldsymbol{i}$). What is the greatest strength you can achieve given the troops available and the money in your coffers? You may assume that there will always be sufficient troops to buy as many as you can afford.

Input Description

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line with two space-separated integers $\boldsymbol{n}$ ($1 \leq \boldsymbol{n} \leq 30{,}000$) and $\boldsymbol{m}$ ($0 \leq \boldsymbol{m} \leq 100{,}000$), where n is the number of troop types and m is the total amount of money you have to spend.

Each of the next $\boldsymbol{n}$ lines will hold three values separated by spaces, representing a type of troop:

$$\boldsymbol{c\ h\ p}$$

Where $\boldsymbol{c}$ ($1 \leq \boldsymbol{c} \leq 100{,}000$) is an integer, which is the cost of that type of troop, $\boldsymbol{h}$ ($0.0 \leq \boldsymbol{h} \leq 1.0$) is a real number, which is the health of that type of troop, and $\boldsymbol{p}$ ($0.0 \leq \boldsymbol{p} \leq 1.0$) is a real number, which is the potency of that type of troop.

Output

Output a single real number, which is the maximum strength you can achieve with your monetary resources. Output this number rounded to exactly 2 decimal places.

Sample Input 1	Sample Output 1
4 100000 300 1 0.02 500 0.2 1 250 0.3 0.1 1000 1 0.1	19436.05

Sample Input 2	Sample Output 2
2 100 1 0.1 1 1 1 0.1	3025.00

Source: 2018 ICPC Southeast USA Regional Programming Contest

G – The Party Animals

Problem Statement

Amira Martinez, the CEO of "The Party Animals, Inc." was approached by the president of the drilling company, WeAreBored Inc., and was asked to help with planning a company party. WeAreBored Inc. has a hierarchical structure; that is, the supervisor relation forms a tree rooted at the president. A supervisor may supervise one or more direct subordinates. The personnel office has ranked each employee with a **conviviality[1] rating**, which is an integer. In order to make the party fun for all attendees, the president does not want both an employee and his or her immediate supervisor to be invited.

Amira is given the tree describing the corporate structure of WeAreBored Inc. and the conviviality rating of every employee in the company. Help Amira to make up a guest list that maximizes the sum of the conviviality ratings of the guests.

Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. The first line contains $N \leq 10^5$, the number of employees in the company. The employees will be numbered 1 through *N*, with the president being employee 1. The next *N* lines will provide information about each of the *N* employees in the order of their employee numbers. The *i*-th line contains information about employee *i* and contains two numbers, the identity of the employee's supervisor and employee *i*'s conviviality rating. The president of WeAreBored Inc. has no supervisor, so her supervisor will be indicated by 0.

Output

Output an integer indicating the maximum possible sum of the conviviality ratings of the employees that Amira can invite to the party.

Sample Input 1	Sample Output 1
7 0 91 1 19 2 17 2 43 2 19 1 29 3 71	224

Sample Input 2	Sample Output 2
9 0 11 1 22 1 33 1 44 2 55 2 66 3 44 3 55 4 66	297

[1] Webster's dictionary: *relating to, occupied with, or fond of feasting, drinking, and good company*

H – Miles to Kilometers

Problem Statement

Few people know that you can use Fibonacci numbers to approximately convert distances from miles to kilometers. It is well known, however, that one mile equals exactly 1.6 km. So, how does the Fibonacci conversion method work? First, these are the first 13 numbers of the Fibonacci sequence to use as a guide:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Suppose you want to convert 5 miles into kilometers. In the Fibonacci sequence, the number following 5 is 8, so we can say 5 miles = 8 km. Unfortunately, this method doesn't work all the time. For example, to convert 89 miles into kilometers, the next Fibonacci number after 89 is 144, which is not the right answer. The correct answer is 89 × 1.6 = 142.4 km, so the Fibonacci method is off by 1.6.

The Fibonacci method sometimes allows multiple ways to convert miles to kilometers. For example, if we want to convert 6 miles into kilometers, we could partition 6 into the sum 5 + 1, convert both 5 and 1 separately to kilometers, and then add the resulting values. For example, 5 miles converts to 8 km and 1 mile converts to 2 km; their sum equals 8 + 2 = 10. Using this method, we can say 6 miles is approximately equal to 10 km. On the other hand, we could partition 6 miles differently, as 2 + 2 + 2, convert 2 miles to 3 km, and produce the sum 3 + 3 + 3 = 9 km. Which answer (9 or 10) is closer to the exact conversion value in kilometers? The exact conversion formula produces 9.6 km, so 10 turns out to be the best Fibonacci conversion we tried.

It is your task to write a program that finds the best conversion from miles to kilometers using the Fibonacci method we described earlier. The best conversion is the one with the smallest error, and you are to print the absolute value of that error.

Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. The first line if input will be the integer T, indicating the number of miles test values to follow. This is followed by T lines of input ($1 < T < 21$). Each case will be a single line containing one integer number M ($0 < M \leq 1000$).

Output Description

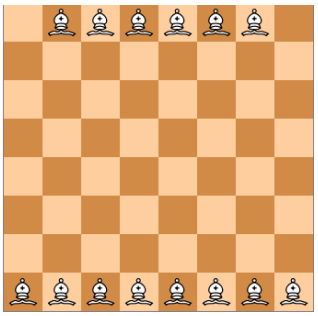
For each test case in the input, your program must print the absolute value of the lowest error for converting the corresponding input to kilometers using the Fibonacci method. Round your answers upward to 2 decimal digits of precision).

Sample Input	Sample Output
3 6 5 12	0.40 0.00 0.20

Source: UVA Online Judge 11780 (mostly rewritten by K. Irvine)

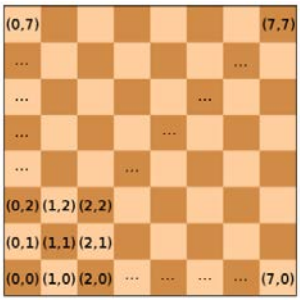
I – Bishop Checking

Problem Statement



The *bishops puzzle* is a classic chess puzzle where you place bishops on an 8 x 8 chess board in such a way that no bishop can attack another bishop. In chess, a bishop can move along any diagonal. One possible solution involving 14 bishops is shown here on the right side of the page. Today, instead of finding solutions to this problem, you need only verify whether a given chess board contains a valid solution.

When referring to specific squares on the board, the bottom-left square is notated (0,0), the x coordinate increases as you move rightward, and the y coordinate increases as you move upward. The figure on the right shows several examples.



Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. Each input begins *N*, an integer that tells you how many bishops will be listed in the input. This is followed by *N* lines, each containing the (x, y) position of a bishop as two integers separated by a single space.

Output Description

The output contains either CORRECT if the given board layout is a solution to the bishops puzzle, and INCORRECT otherwise. Note that the first sample input presented below corresponds to the board shown earlier.

Sample Input 1	Sample Output 1
14 1 7 2 7 3 7 4 7 5 7 6 7 0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0	CORRECT

Source: Adapted from "queens" problem at open.kattis.com

J – Interpreter

Problem Statement

The Basic programming language, invented by professors at Dartmouth college in the 1970s, provided a great way for beginners to learn how to write computer programs. It used an interpreter, a simple utility program that read each program statement on a single line in a source program and executed it before going to the next statement. Your task is to write a simplified interpreter that reads an input program and displays the value generated by the Print statement. Here, for example, is a program with several variables, assignment statements, and a line that prints the value of *N*:

```
A = 2
B = 1
total = A + B
N = 40 + total
Print N
End
```

Input Description

Each input will consist of a single test case. Your program will be run multiple times on different inputs. Input contains three types of statements: (1) Assignment, (2) Print, and (3) End. Print and End statements will each appear only once. An assignment statement begins with a variable name, followed by an = (equal) sign, followed by an expression. An expression consists of a variable name or constant integer, followed by a binary operator such as +, −, *, or /, followed by another variable name or constant integer. (There will be no negative signs or other unary operators.) In division operations, fractional values are truncated to the next lowest integer. A Print statement consists of the word Print followed by a space, followed by a variable. Variable names are 1 to 10 characters, and are case sensitive. Each input line will be no longer than 80 characters, including spaces.

Output Description

For any Print statement in the program, display the variable name, an equal sign (=), and the value of the variable, all on a single line. There should be one space on either side of the equal sign.

Sample Input	Sample Output
A = 2 B = 1 total = A + B N = 40 + total Print N End	N = 43