Lethbridge Collegiate Programming Contest

September 21, 2013

Division I

A: Penny Loser
B: Penny Saver
C: Warmer or Colder
D: Channel Surfing
E: Quaternions

This page is intentionally (almost) blank

University of Lethbridge

A: Penny Saver

LCPC 2013: Division I

Background

(Note that the Background section is exactly the same as in Problem B: Penny Loser)

Recently, the penny has been taken out of circulation in Canada. This has altered the way that cash transactions occur. Here is the Government of Canada's official statement of rounding:

> Under this guideline, cash transactions will be rounded in a fair and transparent manner, as illustrated below:
>
> - Amounts ending in 1 or 2 cents are rounded down to the nearest 10 cents;
> - Amounts ending in 3 or 4 cents are rounded up to the nearest 5 cents;
> - Amounts ending in 6 or 7 cents are rounded down to the nearest 5 cents;
> - Amounts ending in 8 or 9 cents are rounded up to the nearest 10 cents;
> - Amounts ending in 0 or 5 cents remain unchanged.
>
> The calculation of the Goods and Services Tax (GST) on purchases, whether for cash or non-cash transactions, will continue to be calculated to the nearest penny and added to the price. It is only the total cash payment for the transaction that will be rounded.

A GST of 5% of the transaction amount will be added to each bill. Note that when they say "to the nearest penny", they mean 0.5 is rounded up (away from zero). For example, A bill of \$1.02 + GST is \$1.07 and a bill of \$1.30 + GST is \$1.37.

Howard is always looking for interesting ways to save a little bit of money[1], so he is going to exploit this system!

For example, say that an item would normally cost \$5.26 (including tax), then Howard can either pay by credit card (paying the full \$5.26) or he can pay in cash (paying the rounded down value of \$5.25), saving him \$0.01!

Howard is going to the mall with $\$K$ in cash and a credit card that has a credit limit of \$L. This means that the (total) amount of money charged to his credit card may not exceed \$L. He will make n transactions.

Problem

He will process the transactions sequentially in an attempt to save money. (I.e., once he has made the decision on how to pay for a transaction, he will not be able to change it later). When making a purchase, he will choose the first item in this list that applies:

- If he does not have enough cash to pay for this transaction, then he will use his credit card.

[1] Ask him how he saved 3 cents when he ordered from McDonald's!

- If making this purchase would put him above his credit limit, then he will use cash.
- If the bill would be rounded up when using the rules above, then he will use his credit card.
- Otherwise, he will use cash.

You may assume that it is possible to pay for all transactions using this algorithm with the given cash and credit limit.

Input

(Note that the Input section is exactly the same as in Problem B: Penny Loser)

Input will start with a positive integer $T \leq 100$ denoting the number of test cases. Each test case starts with three integers, $0 \leq n \leq 100$, $0 \leq L \leq 1000$ and $0 \leq K \leq 1000$ – whose meanings were given above. Each of the next n lines gives the amount *before tax* of each transaction. All dollar amounts will be in the form *d.cc* (i.e., at least one digit before the decimal and exactly two afterwards).

Output

For each test case, output the case number followed by the total amount of money Howard spent. Follow the style in the sample output.

Sample Input

```
2
2 1000 1000
4.25
5.00
2 1 1
0.82
0.83
```

Sample Output

```
Case 1: $9.70
Case 2: $1.72
```

University of Lethbridge

B: Penny Loser

LCPC 2013: Division I

Background

(Note that the Background section is exactly the same as in Problem A: Penny Saver)

Recently, the penny has been taken out of circulation in Canada. This has altered the way that cash transactions occur. Here is the Government of Canada's official statement of rounding:

> Under this guideline, cash transactions will be rounded in a fair and transparent manner, as illustrated below:
>
> - Amounts ending in 1 or 2 cents are rounded down to the nearest 10 cents;
> - Amounts ending in 3 or 4 cents are rounded up to the nearest 5 cents;
> - Amounts ending in 6 or 7 cents are rounded down to the nearest 5 cents;
> - Amounts ending in 8 or 9 cents are rounded up to the nearest 10 cents;
> - Amounts ending in 0 or 5 cents remain unchanged.
>
> The calculation of the Goods and Services Tax (GST) on purchases, whether for cash or non-cash transactions, will continue to be calculated to the nearest penny and added to the price. It is only the total cash payment for the transaction that will be rounded.

A GST of 5% of the transaction amount will be added to each bill. Note that when they say "to the nearest penny", they mean 0.5 is rounded up (away from zero). For example, A bill of \$1.02 + GST is \$1.07 and a bill of \$1.30 + GST is \$1.37.

Howard is always looking for interesting ways to save a little bit of money[1], so he is going to exploit this system!

For example, say that an item would normally cost \$5.26 (including tax), then Howard can either pay by credit card (paying the full \$5.26) or he can pay in cash (paying the rounded down value of \$5.25), saving him \$0.01!

Howard is going to the mall with \$$K$ in cash and a credit card that has a credit limit of \$L. This means that the (total) amount of money charged to his credit card may not exceed \$L. He will make n transactions.

Problem

He needs to save as much money as possible (he is saving up to buy a new toilet seat for some reason). Can you help him? Can you tell him the minimum total amount of money that he can spend given each of his transactions that day? Each transaction must be paid by either credit card or cash (not a mixture).

You may assume that there exists some way to pay for all transactions with the given cash and credit limit.

[1] Ask him how he saved 3 cents when he ordered from McDonald's!

Input

(Note that the Input section is exactly the same as in Problem A: Penny Saver)

Input will start with a positive integer $T \leq 100$ denoting the number of test cases. Each test case starts with three integers, $0 \leq n \leq 100$, $0 \leq L \leq 1000$ and $0 \leq K \leq 1000$ – whose meanings were given above. Each of the next n lines gives the amount *before tax* of each transaction. All dollar amounts will be in the form *d.cc* (i.e., at least one digit before the decimal and exactly two afterwards).

Output

For each test case, output the case number followed by the minimum total amount of money Howard could spend. Follow the style in the sample output.

Sample Input

```
2
2 1000 1000
4.25
5.00
2 1 1
0.82
0.83
```

Sample Output

```
Case 1: $9.70
Case 2: $1.71
```

University of Lethbridge

C: Warmer or Colder

LCPC 2013: Division I

When you were a kid, do you remember playing the *Warmer or Colder* game? Basically, there was an object hidden in the room that you were supposed to find. So you would close your eyes, and someone would go hide the object. You would then open your eyes and walk to a different position in the room. When you get there, you would stop and the person who hid the object would say "Warmer" if you were closer to the object than before or "Colder" if you aren't any closer to the object than before. You would then make your way to another point in the room, at which point the person who hid the object would tell you "Warmer" or "Colder" based on if you were closer than your previous location. This process would continue until you found the object.

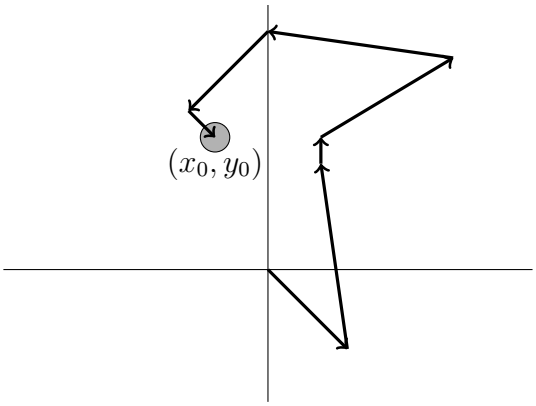

Figure 1: The Sample Input

To simplify things, you may think of the room as a two-dimensional plane. You will always start on the origin, and the object will be hidden at the location (x_0, y_0).

Interactive Input and Output

The input for this problem is different than the normal input for the programming contest. The first line of input will be a positive integer $K \leq 10^{15}$. It it guaranteed that $|x_0| \leq K, |y_0| \leq K$.

The input will come from a program that is interacting with yours. You program must make guesses as to where the object is by outputting 2 integers, x, y. The judging program will then return one of three verdicts: "`Warmer`", "`Colder`" or "`FOUND!`" (without quotes). If you receive one of the first two verdicts, then your program must make another guess. If the last verdict is output, then you have found the object and must produce no more output from your program (and your program should exit). Each integer that is guessed must satisfy $|x| \leq K, |y| \leq K$. Your program may make no more than 1000 guesses.

The image above corresponds to the sample input.

Sample Input/Output

Because this is different than most programming contest questions, the sample input/output is formatted slightly different. Any line starting with "`> `" is assumed to be input for your program and any line without is output from your program. Note that the "`> `" will not appear in the actual input to your program.

```
> 20
3 -3
> Colder
2 4
> Warmer
2 5
> Warmer
7 8
> Colder
0 9
> Warmer
-3 6
> Warmer
-2 5
> FOUND!
```

Techincal Details

For this problem, please ensure that you *flush* your output stream after each entry output. Below are the examples of how to do this for the three supported languages.

C

```
int main() {
  printf("%s\n","Do not forget to flush"); // After every printf,
  fflush(stdout);                          // use fflush(stdout)
}
```

C++

```
int main() {
  cout << "Do not forget to flush" << endl; // After every cout,
  cout << flush;                            // use flush
}
```

Java

```
public class warmcold {
  public static void main(String[] args) {
    System.out.printf("%s\n","Do not forget to flush"); // After every printf,
    System.out.flush();                                 // use flush()
  }
}
```

University of Lethbridge

D: Channel Surfing

LCPC 2013: Division I

Kayleigh has been getting really mad at Darcy lately. In order to turn on our television, there is a very simple process of clicking the red "POWER" button (for the cable converter), then the red "C" button (for the TV), then the blue "B" button (for the surround sound). Then, to get to the correct channel, you may use the guide by pressing "Menu", then "Info" twice (because the "OK" button is broken). Darcy doesn't understand why it is so hard – but to appease her, he has bought a remote with only 5 buttons: a red "POWER" and 4 purple buttons (Labelled "A","B","C" and "D").

Darcy bought the remote because it was extremely simple to use. He has altered the programming of the remote so that the POWER button on the remote turns on all of the required machines. But he isn't THAT nice! He also programmed the 4 purple buttons to be similar to the "Channel Up" button – with a twist.

Pressing the "A" button once is equivalent to pressing the "Channel Up" button a times. That is, the channel number will be increased by a. If, in the process of increasing the channel, you attempt to go above a maximum channel threshold M, then you will *wrap around* to channel 1 and continue increasing (see the examples below for the *wrap around* effect). The "B", "C" and "D" buttons will also have a similar result, but with their own variables b, c and d. Each of these variables will be no more than one million.

For example, if $a = 1, b = 3, c = 4, d = 5$ and $M = 21$, and Kayleigh wants to go from channel 3 to channel 12, she has a few options: ABD ($3 \to 4 \to 7 \to 12$) or DC ($3 \to 8 \to 12$). Then, to go back to channel 3, she could press DADA ($12 \to 17 \to 18 \to 2 \to 3$).

Darcy could also be feeling very mean. He could also make it impossible for her to get from one channel to another. For example, if $a = 2, b = 2, c = 2, d = 2$ and $M = 4$, there is no way to get from channel 1 to channel 2!

So now Kayleigh is upset because she doesn't know if she can even get to a specific channel, and even if she can, she thinks that it takes too much time to get to the channel that she wants to. She needs your help. Can you tell her the minimum number of button presses needed to get from channel x to channel y?

Input

Input will contain multiple test cases. Each test case contains 7 integers split onto two lines. On the first line, there will be three integers, $1 \leq M \leq 1000000, 1 \leq x, y \leq M$. The second line will contain 4 positive integers a, b, c and d.

Output

For each test case, output the minimum number of button presses needed to get from channel x to channel y. If it is impossible, output -1.

Sample Input

```
21 3 12
1 3 4 5
4 1 2
2 2 2 2
```

Sample Output

```
2
-1
```

University of Lethbridge

E: Quaternions

LCPC 2013: DIVISION I

Some of you may know (and some of you may not know), that there is another extension of the real numbers known as *Quaternions* (denoted $\mathbb{H}$). Every quaternion number is of the form $a+bi+cj+dk$, where a,b,c and d are real numbers. The i,j,k are special non-real numbers.

Just like the numbers that you are used to, you can do any of the normal operations on quaternion numbers (addition, subtraction, multiplication, division, etc.). Addition and subtraction are exactly how you would expect, just add (or subtract) the corresponding components together. But with multiplication, things are a little different: it is similar to how you would multiply polynomials: by distributing. Once you've distributed, there is a lot of extra simplification. Those special symbols, i, j and k, multiply together nicely in unexpected ways. The table below shows the result of the $r \cdot c$ (where r is the row and c is the column).

	1	**i**	**j**	**k**
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

For this problem, you will be asked to either add, subtract or multiply two quaternion numbers (division is too hard, so you don't have to worry about it).

Input

The input will start with a number T, the number of test cases to follow. Each test case will appear on exactly one line. On each line, there will be 9 items: `a b c d (op) w x y z`. You may assume that `a`, `b`, `c`, `d`, `w`, `x`, `y` and `z` are integers and that `(op)` is one of "`+`", "`-`" or "`*`" (without the quotes). The first four integers represent the first quaternion number (real part first, then the i-component, then the j-component, then the k-component). The last four integers represent the second quaternion number (in the same format as the first). You may assume that all input integers have absolute value less than 10000.

Output

For each test case, output the simplified result of the computation. Output the answer in the same format as the numbers in the input.

Sample Input

```
3
1 2 3 4 + 5 6 7 8
1 2 3 4 - 5 6 7 8
1 2 3 4 * 5 6 7 8
```

Sample Output

```
6 8 10 12
-4 -4 -4 -4
-60 12 30 24
```

Explanation of Sample Output

Test Case 1:

$$\begin{aligned} & (1+2i+3j+4k)+(5+6i+7j+8k) \\ = \; & (1+5)+(2+6)i+(3+7)j+(4+8)k \\ = \; & 6+8i+10j+12k \end{aligned}$$

Test Case 2:

$$\begin{aligned} & (1+2i+3j+4k)-(5+6i+7j+8k) \\ = \; & (1-5)+(2-6)i+(3-7)j+(4-8)k \\ = \; & -4-4i-4j-4k \end{aligned}$$

Test Case 3:

$$\begin{aligned} & (1+2i+3j+4k)\cdot(5+6i+7j+8k) \\ = \; & 1\cdot(5+6i+7j+8k)+2i\cdot(5+6i+7j+8k)+3j\cdot(5+6i+7j+8k)+4k\cdot(5+6i+7j+8k) \\ = \; & (5+6i+7j+8k)+(10i+12i^2+14ij+16ik)+(15j+18ji+21j^2+24jk)+(20k+24ki+28kj+32k^2) \\ = \; & 5+16i+22j+28k+12i^2+21j^2+32k^2+14ij+16ik+18ji+24jk+24ki+28kj \\ = \; & 5+16i+22j+28k+12(-1)+21(-1)+32(-1)+14k+16(-j)+18(-k)+24i+24j+28(-i) \\ = \; & -60+12i+30j+24k \end{aligned}$$