Money Money Money, Must Be Funny

`money.c`, `money.C`, `money.p`, `money.java`

- **Shopkeeper:** "100 crowns and 80 hellers, please."
- **Customer:** "Here you are." (handing a 200-crown banknote)
- **Shopkeeper:** "Would you have 80 hellers?" (passing back a 100-crown note)
- **Customer:** "No, but here you are." (adding a 1-crown coin)
- **Shopkeeper:** "Sorry, but have only this." (showing a little shiny 50-heller coin)
- **Customer:** "So I need to give you another 30 heller. But I have only 40."
- **Shopkeeper:** "That's ok, here are the remaining 10."

Have you ever experienced similar situations? Paying a precise amount can sometimes be difficult, if the set of available coins and banknotes ("tenders") is limited. The situation above was finally solved: The customer paid 200+1 crowns, got 100+0.50 back, paid another 0.20+0.20, and finally got 0.10 back. This means, 7 tenders had to be exchanged. Sometimes, it may be even more complicated. Your task is to write a program that solves situations like this.

Input Specification

Input contains several tasks to be solved. Each task begins with a line containing one non-negative number: the amount to be paid. Then there is a list of tenders possessed by the customer (the one who pays). Each line in the list contains the tender nominal value (non-negative number), one space, number of tenders of that value (non-negative integer), and the lowercase letter "`x`". The list is terminated by a line containing number "`-1`".

After the first list, there is a list of tenders possessed by the shopkeeper (the one who gets paid). The second list has the exactly same format as the first one.

Then the next task begins. The last task is followed by one more line containing "`-1`".

You may assume that each list will contain at most 100 lines, and nobody will have more than 10 000 units and/or 500 tenders. Nominal values can be arbitrary, they do not need to follow any existing scheme valid in known countries. All numbers that allow non-integer values will be given either as integers or as decimal numbers with one or two digits after the decimal point.

Output Specification

For each task, output one line containing the sentence "*X* `tenders must be exchanged.`", with *X* replaced by the minimal number of tenders that are to be used to pay the required amount. If this is not possible at all, output the sentence "`The payment is impossible.`" instead.

Sample Input

```
100.80
500 1x
200 3x
1.00 10x
0.20 2x
-1
500 10x
200 12x
100 8x
0.10 1x
0.20 0x
0.50 100x
20 2x
-1
200
10 19x
-1
200 1x
-1
-1
```

Output for Sample Input

```
7 tenders must be exchanged.
The payment is impossible.
```