Czech ACM Student Chapter

Charles University in Prague
Slovak University of Technology
University of Žilina
Matej Bel University in Banská Bystrica

Czech Technical University in Prague

Technical University of Ostrava
Pavol Jozef Šafárik University in Košice
Masaryk University
University of West Bohemia

CTU Open Contest 2019

Beer Bill

`bill.c`, `bill.cpp`, `Bill.java`, `bill.py`

Pub bills were introduced into daily life long before computers even existed. People tend to think that once a bill has been paid, it is no more than a pathetic paper scrap not worthy of any more attention. The fact, completely overlooked by established computer science streams, is that the scribblings on the bill represent highly formalized and often quite nontrivial text suitable for many formal language applications and analyses.

A bill consists of lines of characters. Each line is either a priced line or a rake line. A priced line begins with a positive integer — the price of a food or drink item — which is optionally followed by some number of vertical bars. The price of the line is calculated as follows: If the bars are present, the price of the line is equal to the item price multiplied by the number of bars. Otherwise, the price of the line is equal to the price of the item. A rake line contains only vertical bars (similar to rake dents, hence the name rake line), each bar stands for one beer bought by the bill holder. The price of the rake line is the price of one beer multiplied by the number of the bars on the line. The beer price, in this problem, is equal to 42 monetary units. The bill total is the total of prices of all lines on the bill.

We present you with a formal definition of the so-called Raked bill language, as far as we know, the first of its kind in the whole history of computer science. The bills in this problem are expressed in the Raked bill language.

```
<BILL>        ::=  <LINE> | <LINE><BILL>
<LINE>        ::=  <PRICED_LINE><line_break> | <RAKE_LINE><line_break>
<PRICED_LINE> ::=  <PRICE_SPEC> | <PRICE_SPEC><RAKE>
<RAKE_LINE>   ::=  <RAKE>
<PRICE_SPEC>  ::=  <PUB_INTEGER><comma><hyphen>
<RAKE>        ::=  <rake_dent> | <rake_dent><RAKE>
<PUB_INTEGER> ::=  <dig_1_9> | <dig_1_9><DIG_SEQ>
<DIG_SEQ>     ::=  <dig_0_9> | <dig_0_9><DIG_SEQ>
<dig_1_9>     ::=  '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
<dig_0_9>     ::=  '0' | <dig_1_9>
<rake_dent>   ::=  '|'          // ascii character 124
<comma>       ::=  ','          // ascii character 44
<hyphen>      ::=  '-'          // ascii character 45
<line_break>  ::=  LF           // ascii character 10, line feed
```

In the language specification above, the actual characters which appear on the bill are enclosed in single quotation marks to distinguish them from the other parts of the specification. The symbol `//` introduces a single line comment, it is not part of the language definition.

Input Specification

The input contains a nonempty sequence of lines which specify a bill. Each input line is either a priced line or a rake line. A priced line starts with a positive integer, not exceeding 1 000, followed immediately by a comma and a minus sign. Optionally, the minus sign is followed by a nonzero number of vertical bars. A rake line contains nonzero number of vertical bars and no other symbols. In the whole bill, the vertical bar is represented as `'|'`, ascii character 124. Each line contains at most 1 000 characters, there are no blanks on the line. There are at most 1 000 input lines. The input does not contain any empty line. All prices are expressed in the same monetary units.

Output Specification

Output a single number — the bill total rounded up to the nearest 10s. The number should be in the `<PUB_INTEGER>` format, that is, it should be immediately followed by a comma and a minus sign.

Sample Input 1

```
||||
123,-|||
```

Output for Sample Input 1

```
540,-
```

Sample Input 2

```
|||
12,-|
|||
12,-||
10,-|
```

Output for Sample Input 2

```
300,-
```

Sample Input 3

```
|
8,-|
```

Output for Sample Input 3

```
50,-
```