# Problem I – Invoice Cost Reduction.

*Author*: Moroni Silverio

Jaime has been learning web development for a while, sometimes he loves it, sometimes he hates it. Recently he learned about a technology called serverless computing in which he doesn't have to pay for a server running 24/7 which is not being used every hour of the day.

This technology has the concept of "pay as you go", which means he only pays every time a section of code is executed, if nothing is executed he doesn't have to pay anything.

According to the invoices he receives from the serverless company which hosts his code, he is being billed for the execution time of his code.

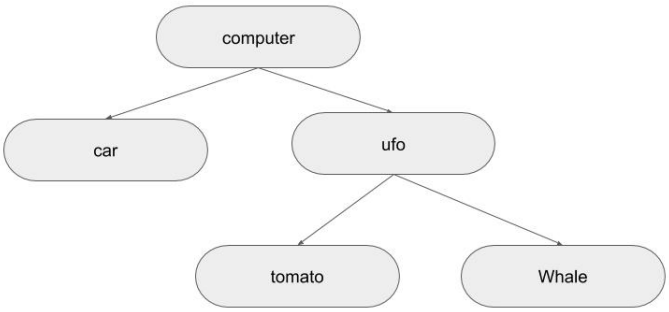
Jaime has modernized his little store and now he is selling via the internet and has implemented a function that retrieves the prices of the products he is selling.

Originally he had the product's prices in a list, but finding their prices was time consuming. So he decided to store the prices in a binary search tree.

A binary search tree goes as follows: Give a node in the tree (root for example), the value stored in it [the name of a product] is lexicographically greater than any other value of any other node of the left sub tree and that same value is lexicographically smaller than any other value of any other node of the right sub tree.

The execution time of the function that retrieves the price of a product depends on the number of comparisons made in order to find the product in the tree.

For example, if the products for sale are *computer*, *car* $ufo$, *tomato* and *whale* and the following tree is made:




Two comparisons would be made to find the ufo's price, three comparisons to find tomato's price, and only one comparison to find the computer's price.

Jaime also loves data mining and has stored the frequencies in which the price of each product is asked. Jaime has created a function to calculate the expected invoice to be paid and is defined as follows:

$$\sum_{product \in Products} Frequecy(product) * Comparisons\_to\_find(product)$$

Jaime has discovered that different binary search trees may lead to different expected invoices. Can you help him find the best tree that minimizes the expected invoice? Of course you can.

## Input

Input begins with a number $N$ ($1 \leq N \leq 100$), the number of products Jaime sells over the Internet, $N$ lines will follow, each describing a product for sale in the following manner: a nonempty string $s$ representing the name of the product (the name is composed of at most 100 characters in lower case $[a - z]$ and no two product have the same name), then, in the same line two integer numbers $p$ and $f$, ($1 \leq p \leq 10^5$) and ($1 \leq f \leq 10^7$), the price and the frequency of the mentioned product.

## Output

Output the minimum expected invoice achievable.

| **Sample input 1** | **Sample output 1** |
|---|---|
| 5<br>computer 50 50<br>car 800 40<br>ufo 90 40<br>tomato 1 30<br>whale 100 30 | 390 |