Problem Statement for **Gumpler Travels**

Problem Statement:
There is a trading company (Gumpler Ltd.) which provides its customers travel across different countries. In a particular situation, the company has chosen a travel encompassing 4 different countries (say A, B, C and D). Each country has 1000 different locations (numbered from 0 to 999).

A customer requests a travel from a company (we call it a travel address) in the form of a concatenation of four integers from 0 to 999,inclusive, in decimal notation with no extra leading zeroes, separated by periods ('.'). For example, the following two strings are valid travel addresses :

```
66.37.210.86
123.456.789.0
```

In addition, in a travel address, there is a way to represent some addresses by using a wildcard character '*'. Each '*' character replaces one of the four integers in the address, and represents all integers between 0 and 999, inclusive. For example, "23.4.*.8" represents 1000 travel addresses: "23.4.0.8", "23.4.1.8", ..., "23.4.999.8", and "*.4.*.8" represents 1000000 travel addresses.

Gumpler Ltd. has received requests from some customers to allocate them a travel. i-th customer requests travel addresses represented by request[i], and the customer promises to pay price[i] dollars for each requested address.

Each travle address can be allocated to at most one customer, and each customer does not require all the travel addresses he requested. Assume that Gumpler Ltd. owns all the travel addresses that have been requested by the customers.

Return the maximum amount of money Gumpler Ltd. can gain.

Input:
The first line inputs N where N is the number of customers.
For each customer, there are two input lines, first line contains the travel address requested and the second line contains the price customer will pay for the request. (Format will be clear by the examples).

OutPut:
1 line, output the maximum amount gained.

Constraints:
number of customers are between 1 and 50, inclusive.
price will be between 1 and 1,000,000, inclusive.
N will range from 1 to 10.
Compilation time: 10 seconds,
Execution time: 5 seconds.
Memory usage: 256 MB.

Note : Do not use hard coding as it will run out of time constraints on the sample input. Also, **use long integer for the final amount**.

Examples:

Input:

```
1
66.37.210.86
47
```

Output:

```
47
```

An optimal way is to sell travel address "66.37.210.86" to the 0-th customer (the only customer in this case). Gumpler Ltd. would gain 47 dollars.

Input:

```
3
0.0.0.*
1
0.0.0.3
3
0.0.0.5
9
```

Output:

```
1010
```

Gumpler Ltd. is going to sell 1,000 addresses "0.0.0.0", ..., "0.0.0.999" to the customers.

An optimal way is to sell "0.0.0.3" to customer 1, "0.0.0.5" to customer 2, and the other 998 addresses to customer 0. The company would gain 3*1+9*1+1*998=1010 dollars.

Input:

```
4
127.0.0.1
999999
*.0.0.*
629851
*.*.255.255
294016
192.68.*.*
438090
```

Output:

1361957076132