# **Tries: Contacts**



We're going to make our own *Contacts* application! The application must perform two types of operations:

- add name, where name is a string denoting a contact name. This must store name as a new contact in the application.
- 2. **find partial**, where **partial** is a string denoting a partial name to search the application for. It must count the number of contacts starting with **partial** and print the count on a new line.

Given n sequential add and find operations, perform each operation in order.

#### **Input Format**

The first line contains a single integer, n, denoting the number of operations to perform. Each line i of the n subsequent lines contains an operation in one of the two forms defined above.

#### **Constraints**

- $1 \le n \le 10^5$
- $1 \le |name| \le 21$
- $1 \leq |partial| \leq 21$
- It is guaranteed that *name* and *partial* contain lowercase English letters only.
- The input doesn't have any duplicate *name* for the *add* operation.

#### **Output Format**

For each find partial operation, print the number of contact names starting with *partial* on a new line.

## **Sample Input**

```
4
add hack
add hackerrank
find hac
```

# **Sample Output**

2 0

### **Explanation**

We perform the following sequence of operations:

- 1. Add a contact named hack.
- 2. Add a contact named hackerrank.
- 3. Find and print the number of contact names beginning with  $\frac{hac}{hac}$ . There are currently two contact names in the application and both of them start with  $\frac{hac}{hac}$ , so we print  $\frac{2}{a}$  on a new line.
- 4. Find and print the number of contact names beginning with  $\frac{hak}{hak}$ . There are currently two contact names in the application but neither of them start with  $\frac{hak}{hak}$ , so we print 0 on a new line.