Contacts

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

1. `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.

Example
Operations are requested as follows:

```
add ed
add eddie
add edward
find ed
add edwina
find edw
find a
```

Three add operations include the names 'ed', 'eddie', and 'edward'. Next, $\texttt{find ed}$ matches all 3 names. Note that it matches and counts the entire name 'ed'. Next, add 'edwina' and then find 'edw'. 2 names match: 'edward' and 'edwina'. In the final operation, there are 0 names that start with 'a'. Return the array $[3, 2, 0]$.

Function Description

Complete the *contacts* function below.

contacts has the following parameters:

- *string queries[n]:* the operations to perform

Returns

- *int[]:* the results of each find operation

Input Format

The first line contains a single integer, n, the number of operations to perform (the size of $queries[]$). Each of the following n lines contains a string, $queries[i]$.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq \texttt{length of } name \leq 21$
- $1 \leq \texttt{length of } partial \leq 21$

- $name$ and $partial$ contain lowercase English letters only.

- The input does not have any duplicate $name$ for the add operation.

Sample Input

```
STDIN           Function
-----           --------
4               queries[] size n = 4
add hack        queries = ['add hack', 'add hackerrank', 'find hac', 'find hak']
add hackerrank
find hac
find hak
```

Sample Output

```
2
0
```

Explanation

1. Add a contact named `hack`.

2. Add a contact named `hackerrank`.

3. Find the number of contact names beginning with `hac`. Both name start with `hac`, add 2 to the return array.

4. Find the number of contact names beginning with `hak`. neither name starts with `hak`, add 0 to the return array.