Pet Food Distribution

Objectives

Give practice with tries in C

Story

You managed to secure the food taken by the raccoons and foil BTS’s plan to monopolize pet stores. It’s all over; you’re finally done…

But it’s never that easy. Now you need to work on distributing all the food you reclaimed from the raccoons. You are working on a site that can manage the food given to parts of your animal community. Your employer KAT pet shop is afraid that there might be some pet owners that are abusing the pet food distribution service by getting more than their fair share. You've been tasked with writing a program to help determine if there are any groups of pet owners that have substantially benefited more than others. Hopefully, no one from BTS is trying to get food to resell it on the market…

You have devised a questionnaire to help break the animal community into components. Families have to register with you to receive the pet food by answering enough questions from the questionnaire to uniquely identify them. The questions of the questionnaire starts with coarse level information such as “how many pets do you have?” and “in which region of the country are you located?” into finer level information such as “what breed of beagle do you own?” and “how many spots does your cow have?”. Each answer to these questions is represented as a single character that could have 1 of 32 values (‘0’ through ‘9’ and ‘a’ through ‘v’). The drawback is that some pet owners may move or get new pets, and the answer to these questions can change over time.

Problem

Develop a tracker that will support calculating how much each person has taken, someone changing their questionnaire, and queries of how much food has been given to how many people that have answered the questionnaire in a particular way.

Input

Following this will be a series of command in 1 of the following 5 formats,

- ADD ***R A***
  - ○ Which represents a family that submitted a questionnaire response of ***R*** receiving ***A*** food.
- CHANGE ***R N***
  - ○ Which represents a family that currently has a questionnaire response of ***R*** changing their questionnaire response to ***N***.
- SUM ***R***
  - ○ Which represents a request to find the number of families that have (or could have, if not responded to) answered the first group of questions from the questionnaires with the response(s) of ***R***.
- QUIT
  - ○ Which represents the end of the commands.

For bounds,

- The length of ***R*** and ***N*** will be at least 1 and no more than 100,000
- The value of ***A*** will be at most 10,000

Output

For each SUM command, the program should output a line containing the minimum and maximum total amount of food received by families that currently match (or could currently match) the response(s) provided in the first group of questions from the questionnaire. The minimum and maximum should be separated by a space.

Samples

Input	**Output**
ADD ab1 8 ADD abc 6 ADD a1b 7 ADD a 5 SUM ab SUM a1 QUIT	14 19 7 12
ADD a 8 ADD a 6 SUM ab QUIT	0 14
ADD a 10 SUM a CHANGE a b SUM a QUIT	10 10 0 0

Case Description

In the first case there were 4 families:

- ab1 - 8 food
- abc - 6 food
- a1b - 7 food
- a - 5 food

After receiving all the food, the program was asked how much families that responded with `ab` received. The response `ab` could include families that responded with `ab1`, `abc`, and `a`. The family that responded with `a` is not guaranteed to be in that group, so its count will not be in the minimum, but will be included in the maximum. The minimum is 8 + 6 (14), and the maximum is 8 + 6 + 5 (19).

The response `a1` could include families that responded with `a1b` and `a`. The family that responded with `a` is not guaranteed to be in that group, so its count will not be in the minimum, but will be included in the maximum. The minimum is 7, and the maximum is 7 + 5 (12).

In the second case there was one family. That family received 8 units of food and later 6 units of food. A `SUM` command was issued for response `ab`. The minimum was 0, because the only family might respond with a non '`b`' response to the second question, but the maximum is 14, because that family could have answered '`b`' to the second question, but did not.

In the third case, a single family was given 10 units of food. The first `SUM` command included that family. After that `SUM` command that family changed their response from '`a`' to '`b`', the second `SUM` command looking for that same family now finds no family and no food.

Hints

Node Struct: The instructor’s solution used the following Node struct,

```
struct Node {
   int subTrieAmount;
   int myAmount;
   Node * children[32];
};
```

Character Mapping: Map the characters of ‘0’ through ‘9’ into 0 through 9. Map the characters ‘a’ through ‘v’ into 10 through 31.

Recursively Branching: Don’t try to loop through a full subtrie ever (unless deleting the trie). Such behavior is slow. Recursion is not needed with the exception of deleting the trie.

Change Command: Consider using an addition of a negative amount to remove a family when changing their response.

Min VS Max: The minimum should only include families that answered up to (and after) the end of the requested response. The maximum should include families that answer up to (and after) the end of the requested response, but it should also include families that answer all prefixes (and no more) of the given response.

Grading Criteria

- Read/Write from/to **standard** input/output (e.g. scanf/printf and no FILE *)
    - 5 points
- Good comments, whitespace, and variable names
    - 15 points
- No extra input output (e.g. input prompts, “Please enter the number of cats:”)
    - 5 points
- Use a Trie
    - 5 points
- Each function should be runtime O(length of Response)
    - 5 points
- Map characters to integers
    - 5 points
- Remove the old sum information when changing a families response
    - 5 points
- Process commands until QUIT
    - 5 points
- Programs will be tested on 10 cases
    - 5 points each

No points will be awarded to programs that do not compile using “gcc -std=gnu11 -lm”.

Sometimes a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem implement a trie. ***Without this, programs will earn at most 50 points!***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.