Greedy Gift Givers

USACO Training Pages

Problem Statement

A group of $NP(2 \leq NP \leq 10)$ uniquely named firneds has deicded to exchange gifts of money. Each of these friends might or might not give some money to some or all of the other friends (although some might be cheap and give to no one). Likewise, each friend might or might not recieve money from any or all of the other friends. Your goal is to deduce how much more money each person recives than they give.

The rules for gift-giving are potentially different than you might expect. Each person does to the bank (or any other source of money) to get a certain amount of money to give and divides this money evenly among all those to whom he or she is giving a gift. No fractional money is available, so dividing 7 among 2 friends would be 3 each for the friends with 1 left over - that 1 left over goes into the giver's "account". All the participants' gift accounts start at 0 and are decreased by money given and increased by money recieved.

In any group of friends, some people are more giving than others (or at least may have more acquaintances) and some people have more money than others.

Given:

- a group of friends, no one of whom has a name longer than 14 characters
- the money each person in the group spends on gifts
- a (sub)list of friends to whom each person gives gifts

Determine how much money each person ends up with.

Input Format

Line	Contents
1	A single integer, NP
$2..NP+1$	Line $i+1$ contains the name of group member i
$NP+2..end$	NP groups of lines organized like this: The first line of each group tells the person's name who will be giving gifts. The second line in the group contains two numbers: • The amount of money in the range 0..2000 to be divided into gifts by the giver. • $NG_i (0 \leq NG_i \leq NP)$, the number of people to whom the giver will give gifts. If NG_i is nonzero, each of the next NG_i lines lists the name of a recipient of a gift, recipients are not repeated in a single giver s list.

Sample Input

```
5
dave
laura
owen
vick
amr
dave
200 3
laura
owen
vick
owen
500 1
dave
amr
150 2
vick
owen
laura
0 2
amr
vick
vick
0 0
```

Output Format

The output is NP lines, each with the name of a person followed by a single blank followed by the net gain or loss ($M_f - M_i$, where M_f is the final amount of money and M_i is the initial amoutn of money) for that person. The names should be printed in the same order they appear starting on line 2 of the input.

All gifts are integers. Each person gives the same integer amount of money to each friend to whom any money is given, and gives as much as possible that meets his constraint. Any money not given is kept by the giver.

Sample Output

dave 302
laura 66
owen −359
vick 141
amr −150

Output Explanation

Round	Dave	Laura	Owen	Vick	Amr
0	0	0	0	0	0
1	$0-200+2=-198$	$0+66=66$	$0+66=66$	$0+66=66$	0
2	$-198+500=302$	66	$66-500=-434$	66	0
3	302	66	$-434+75=-359$	$66+75=141$	$0-150=-150$
4	302	66	−359	141	−150
5	302	66	−359	141	−150

In rounds 4 and 5, no money was distributed, therefore the values stay the same.

Approach

The easiest way to solve the problem is to simulate the events. Each person in the testcase will be stored along with their gift account balance (starts at 0). As each query comes, subtract the total amount N from the giver's gift account and add $\frac{N}{M}$ to each of the M gift recievers. Then, add N mod M back to the giver's account.

Technical Details

To keep track of the gift account balance for each person, a map can be used.

Code

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
   freopen("gift1.in", "r", stdin);
   freopen("gift1.out", "w", stdout)
   int n;
   cin >> n;
   // Keep track of the order in which queries come (for output)
   vector<string> order;
   // Gift account balance for each person
   map<string, int> people;
   for (int i = 0; i < n; ++i) {
      string name;
      cin >> name;
      order.push_back(name);
      people.insert(pair<string, int>(name, 0));
   }
   for (int i = 0; i < n; ++i) {
      string giver;
      cin >> giver;
      int total, num_people;
      // Check if any money will be distributed
      if (num_people > 0 && total > 0) {
         map<string, int>::iterator it = people.find(giver);
         // Subtract the amount given from the giver's account and add the remaining
             back
         it->second = (it->second - total) + (total % num_people);
         for (int j = 0; i < num_people; ++j) {
            string person;
            cin >> person;
            map<string, int>::iterator it = people.find(person);
            // Add the money given to the reciever's account
            it->second += total / num_people;
         }
      }
   }
   for (int i = 0; i < n; ++i) {
      map<string, int>::iterator it = people.find(order[i]);
      cout << it->first << " " << it->second << endl;
   }
}
```

Credits

- USACO Training Pages Problem Link
- Author's code repository