Final Test

2018-01-10

Task: Special days

We have a list of special days (birthday, name day, holidays etc.) of our friends (they have unique names).

Write a program that solves the following subtasks:

a) Count and list all the special days (dates) that belong to a given person.

b) Count and list all of our friends (each person should appear in the list only once).

c) Create a summary about all the special days of each month.

d) In which month are there the most special days, and how many special days are there in that month?

The *first line* of the **standard input** contains the name you should use for task a). The *second line* contains the number of special days (1≤N≤50). Then comes the description in the next 2*N lines, in ascending order of the dates. We have the following information about a special day: the friend's name who celebrates that day (1st line, it is not empty, but can consist of more than one word) and the date of the day given by the number of the month (1≤month≤12) and the number of the day (1≤day≤31).
The *first line* of the **standard output** should contain the solution to task a). It should start with the number of special days, then comes the list of dates (month and day) in ascending order, separated by spaces.
The *second line* should contain the solution to task b). It should also start with the number of friends, then comes the list of names that should be in the order of appearance in the input. Separate the names from each other with a comma.
The *third line* should contain the solution to task c), i.e. 12 numbers that are the count of special days in each month, separated by spaces.
The *fourth line* should contain the solution to task d), i.e. the month and the count of special days. If there is more than one solution, give the month that comes first in the year.

Input (`keyboard`)

#	Line content [*explanation*]
1.	`L Piroska` [the person's name]
2.	`7` [*N=7*]
3.	`N Veronika` [*1st person's name*]
4.	`1 13` [*1st person's special day*]
5.	`L Piroska` [*2nd person's name*]
6.	`1 18` [*2nd person's special day*]
7.	`Sz Peter` [*3rd person's name*]
8.	`6 29` [*3rd person's special day*]
9.	`Sz Domonkos` [*4th person's name*]
10.	`8 4` [*4th person's special day*]
11.	`Sz Peter` [*5th person's name*]
12.	`8 4` [*5th person's special day*]
13.	`Sz Domonkos` [*6. person's name*]
14.	`8 25` [*6th person's special day*
15.	`L Piroska` [*7th person's name*]
16.	`12 24` [*7th person's special day*]

Output (`screen`)

#	Line content [*explanation*]
1.	`2 1 18 12 24` [*task a)*]
2.	`4 N Veronika,L Piroska,Sz Peter, Sz Domonkos` [*task b)*]
3.	`2 0 0 0 0 1 0 3 0 0 0 1` [*task c)*]
4.	`1 2` [*task d)*]

So you should write 4 lines to the standard output, in the order of the subtasks. If a subtask needs writing several pieces of information to the standard output, pay attention to using the appropriate separator (spaces or commas). If you can't solve one of the subtasks, then you should write an empty line instead of the solution! You should not write anything else to the standard output! The uploaded version of your program should not contain a statement to wait for a keyboard press (the evaluation system can't press any keys…).

We only assess programs that aim at solving the actual problems. All the tries that just want to test the Biro system will be evaluated as 0 point (even if you have worked hard with it ☺).

Evaluation

Based on 10 test files: **10*(2+3+2+3)=10*10=100 points**

Minimum points required to pass this exam: **50 points**

Access to the evaluation system: https://biro.inf.elte.hu/faces/login.xhtml

User id: the identifier you use to log in to lab computers.

Password: the password you use with user id at lab computers.

Menu items:

- When you submit (`SUBMIT`), you should select the task name and the programming language (cpp)! The file name can be anything.
- You can see the evaluation of all your submission on the `RESULT` page.
- You can download the task description and the sample input/output files (which contain a small and a large input test case) on the `DOWNLOAD` page.
- On the `BACKUP` page, you can download any of your previous submissions.

Time limit: 0.1 seconds, if your program runs longer than that, it means you have an infinite loop.

You program should end with `return 0`!

You can use the following include lines in your program:

- #include <iostream>
- #include <stdlib.h>
- #include <cmath>

Some common error messages of the evaluation system:

- Compile error: the compilation was unsuccessful
- Time limit error: your program exceeded the time limit (you might have an infinite loop in your program)
- Output format error: the format of the output does not correspond to the task description
- Wrong output: the output is not right

If your output is right, you will see `OK` next to the test case.