Final Test

2018-01-24

Task: Candy Shopping

There are M factories in the country who make candies for Christmas. They make K types of candies. One type of candy can be produced at several different factories. We have data about a store that sells N types of candies.

Write a program that solves the following subtasks:

a) Which is the cheapest candy that you can buy in this store?

b) From which factory comes the highest number of different types of candy?

c) How much does the most expensive candy of each factory cost?

d) How many different types of candies can be bought in this store?

e) Which candy types can be bought from only one factory?

The *first line* of the **standard input** contains the number of candy types that are sold in the shop ($2 \le N \le 10000$), the number of factories ($1 \le M \le 100$), and the number of different candy types ($1 \le K \le 100$). The next N lines contains the identifier of the factory ($1 \le F_i \le M$), the identifier of the candy type ($1 \le F_i \le K$) and the price of that candy type from that factory ($1 \le Price_i \le 10000$)

The *first line* of the **standard output** should contain the factory ID and the type ID of the cheapest candy in the shop. If there is more than one solution, it does not matter which one you write to the output.
The *second line* should contain the factory ID from which the highest number of different types of candy are sold in the store. If there is more than one solution, the output should be the factory with the lowest identifier.
The *third line* should contain the solution to task c). It should start with the count of different factories (G) that sell candies in this store. This is followed by G pairs of numbers: the identifier of the factory and the price of the most expensive candy of that factory. You should list the factories in *ascending order* of factory ID.
The *fourth line* should contain the count of different candy types that can be bought in this store.
The *fifth line* should contain the count of candy types that can be bought only from one factory, and the identifiers of those candy types.

#	Input (`keyboard`) Line content [*explanation*]
1.	7 5 6
2.	1 4 2000
3.	3 1 1500
4.	5 1 3000
5.	5 4 4000
6.	5 3 4000
7.	1 1 1000
8.	4 2 1000

#	Output (`screen`) Line content [*explanation*]
1.	1 1
2.	5
3.	4 1 2000 3 1500 4 1000 5 4000
4.	4
5.	2 2 3

Illustration to help you understand the task:

	Type 1	Type 2	Type 3	Type 4
Factory 1	1000			2000
Factory 2				
Factory 3	1500			
Factory 4		1000		
Factory 5	3000		4000	4000

So you should write 5 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. 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*(1+2+2+3+2)=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** are entitled to 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.