CS1020 Sit-in Lab 03 A – Process Scheduling
Semester 2 AY2014/2015

IMPORTANT NOTE: You are to use the appropriate Java API (LinkedList/Queue/Stack) for this problem. You do not need to implement LinkedList/Queue/Stack yourself.

Problem Description

Part 1

We want to simulate the process scheduling service of an operating system. At any point of time, a user may start to execute a new process or may want to kill a process. A process can have different priorities. We consider priority of a process to be represented by an integer and the higher the value, the higher is the priority. For example, a process with priority 3 has higher priority over another process with priority 2.

We consider that in our scheduling system we have the following 2 operations:

1. **execute <*processName*> <*priority*>**
   - **a.** *processName* is a single word and is the name of the process.
   - **b.** *priority* is the priority of the process and is a positive integer.
2. **kill <numberOfProcessesToKill>**

   The user will specify the number of processes he/she wants to kill. Among the currently executing processes, the process which has the **lowest priority** and has **started the earliest** among its priority group will be killed. The number of processes the user wants to kill will not be more than the number of processes currently executing.

You can assume all the process names to be unique.

Input

The first line of input is a positive integer *N* indicating the number of operations that will follow. The next *N* lines are the operations to be performed. Consider the following example. (Note that comments on the right are for explanatory purpose and they do not appear in the input.)

`7`	There are 7 operations
`execute Chrome 5`	Start process Chrome with priority 5
`execute Word 3`	Start process Word with priority 3
`execute Netbeans 4`	Start process Netbeans with priority 4
`execute Paint 1`	Start process Paint with priority 1
`kill 2`	To kill 2 processes
`execute Excel 2`	Start process Excel with priority 2
`execute Flash 5`	Start process Flash with priority 5

Output

After each operation the program should display the currently executing processes, from lowest priority to highest priority, and for processes with the same priority they should be displayed in the order of their execution start time.

If there are no processes currently running, print out ---.

Output for each operation should be followed by a blank line. For the above input the output is shown on the right:

```
Chrome,5

Word,3
Chrome,5

Word,3
Netbeans,4
Chrome,5

Paint,1
Word,3
Netbeans,4
Chrome,5

Netbeans,4
Chrome,5

Excel,2
Netbeans,4
Chrome,5

Excel,2
Netbeans,4
Chrome,5
Flash,5
```

Important notes:

- Please check the input and output files in your account.
- 7 test cases will be used for Part 1, and 3 test cases for Part 2 (see below).

Part 2

Now consider a more realistic situation, where the number of processes that can be executed at a particular point of time is limited. For this problem let us assume that the maximum number of processes that the operating system can handle is **5.** Now modify your above program such that, when a new execution request arrives, if the priority of the arriving process is greater than any currently executing process then the lowest priority process is **killed** and the new process starts executing. However, if the priority of arriving process is less than or equal to all the processes currently executing then no change occurs to the process list.

Input

Consider the following input:

`7`	There are 7 operations
`execute Chrome 5`	Start process Chrome with priority 5
`execute Word 3`	Start process Word with priority 3
`execute Netbeans 4`	Start process Netbeans with priority 4
`execute Paint 1`	Start process Paint with priority 1
`execute Excel 2`	Start process Excel with priority 2
`execute Flash 5`	Start process Flash with priority 5. This will kill Paint and start Flash.
`kill 2`	To kill Excel and Word

Output:

The output for the above input is shown below

```
Chrome,5

Word,3
Chrome,5

Word,3
Netbeans,4
Chrome,5

Paint,1
Word,3
Netbeans,4
Chrome,5

Paint,1
Excel,2
Word,3
Netbeans,4
Chrome,5

Excel,2
Word,3
Netbeans,4
Chrome,5
Flash,5

Netbeans,4
Chrome,5
Flash,5
```

Skeleton Program

You are provided with a skeleton program **ProcessScheduling.java** which is shown on the next page. Do not create any other file. Do not change the name of the **ProcessScheduling** class.

The class **Process** has been done for you. There is no need to change this class. However, you may change the skeleton code in the **ProcessScheduling** class, add methods and classes in any way you deem fit.

```
import java.util.*;

/* Class representing a process */
class Process {
   private String processName;
   private int priority;

   public Process(String processName, int priority) {
       this.processName = processName;
       this.priority = priority;
   }

   public String getProcessName() {
       return processName;
   }

   public void setProcessName(String processName) {
       this.processName = processName;
   }

   public int getPriority() {
       return priority;
   }

   public void setPriority(int priority) {
       this.priority = priority;
   }

   public String toString() {
       return new String(processName + "," + priority);
   }
}

/* Class representing process scheduling service */
public class ProcessScheduling {
   private ... processList;
   private int maxProcesses;



}
```