Printing

Mr. Panda has been asked by the School of Computing (SoC) to implement a printing queue. However, some print jobs have more priority than others. For example, printing exam papers will be considered a high priority print job. Hence, Mr. Panda has been asked to implement the printing queue using a **Priority Queue** that will always print the highest priority print jobs first. For this printing queue, each print job that is added will have a priority level. **Jobs with a higher priority level will be printed first.** However, multiple print jobs may have the same priority level. In that case, these jobs can be printed in any relative order.

In addition to that, sometimes processing the print jobs one at a time can be very slow. Thus, SoC has also asked Mr. Panda to add an operation to the printing queue that immediately prints all of the jobs with the **same priority as the highest priority job**. (In other words, all the print jobs with the **maximal** priority.)

Overall, Mr. Panda wants to simulate a **Priority Queue** that can support the following operations:

Operation	Description
`ADD [priority]`	A print job with priority **[priority]** should be added to the printing queue.
`PRINT`	The printer should print the **highest priority job**. If there are multiple jobs with the same highest priority, the printer will arbitrarily pick any **one** of them. The command should then output the *priority* of that job.
`PRINTALL`	The printer should print **all the jobs with same priority as the highest priority job**. The command should then output the *number of jobs printed* and the *priority* of those jobs. Since these jobs all have the same priority, you only need to output the priority once.

In addition, Mr. Panda wants to know the status of all the print jobs at the end of all the operations. Thus, he wants you to list the priority of all the print jobs **left in the printing queue** in **non-decreasing** order.

Input
The first line of input contains an integer **Q**. **Q** lines will follow, representing an operation each. The operations should be executed in order and the format would be as described in the table above. (See sample)

Output
For each **PRINT** operation, output the priority of the job printed.
For each **PRINTALL** operation, output the number of jobs printed and the priority of the jobs. Refer to the sample testcase for more details.
At the end of all **Q** operations, output the priority of all the **unprinted** jobs in **non-decreasing** order. Add a single space between two consecutive jobs. **Do not print a space after the last job.** Instead, remember to print an end-line character at the end of the output.

Limits
- $1 \leq \mathbf{Q} \leq 100{,}000$
- All the priorities will range from 1 to 10^9 inclusive.
- It is guaranteed there is at least one print job when a **PRINT** or **PRINTALL** operation happens.
- It is guaranteed there will be at least one print job at the end of all the operations.

Sample Input (**printing1.in**)	Sample Output (**printing1.out**)
12 ADD 5 ADD 4 ADD 5 ADD 6 PRINT PRINT ADD 3 ADD 7 ADD 4 ADD 5 PRINTALL PRINTALL	6 5 1 7 2 5 3 4 4

Explanation

After first 4 **ADD** operations, the printing queue will be [4, 5, 5, 6] in non-decreasing priority order. The next 2 **PRINT** operations will print a job with priority 6 and then 5, leaving [4, 5] in the queue. After the next 4 **ADD** operations, the printing queue will be [3, 4, 4, 5, 5, 7]. The next **PRINTALL** operation will print 1 job with priority 7. The **PRINTALL** operation after that will print 2 jobs with priority 5. At the end, the jobs with priority [3, 4, 4] will be left in the printing queue.

Notes:

1. You should develop your program in the subdirectory **ex3** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **2 sec** on Codecrunch.

Skeleton File – Printing.java

You are given the below skeleton file `Printing.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```
/**
 * Name         :
 * Matric. No   :
 * PLab Acct.   :
 */

import java.util.*;

public class Printing {
    private void run() {
        //implement your "main" method here
    }

    public static void main(String[] args) {
        Printing newPrinting = new Printing();
        newPrinting.run();
    }
}
```