IVLE

You have recently been hired by the International College of Panda Coders (ICPC). Being a very good programmer, the school has hired you to help them code their Integrated Virtual Learning Environment (IVLE).

This school has a list of students and modules. Each module is worth some number of modular credits (MCs) and has a maximum number of students that can be registered to avoid overworking the professor. One at a time, each student will register for a particular module. If there are still vacancies, the student is registered and the MC of the module is added to the student's total MC count. A student can also be removed from a module. It is guaranteed that no two students will have the same name.

For data processing purposes, the ICPC also needs the IVLE to be able to process certain queries about the students. When a student logs into the IVLE, the student would want to see all the modules he/she is registered for and the total number of MCs allocated to him/her.

Furthermore, the school also wants to know what is the current total number of students that are registered into at least one module and which of its students have the highest workload.

Since you are a very good programmer, the school has employed you to help them program an IVLE system that is able to process all of the queries listed above.

Good luck!

Input

The first line of input consists of two space-separated integers N (1 <= N <= 10) and Q (5 <= Q <= 1000), representing the number of modules and the number of queries being asked.

The next N lines will contain information about the modules. The first string of each line will contain the module code, which is a string of alpha-numeric characters at least 2 characters long and at most 10 characters long. The next 2 positive integers represent the number of MCs the module is worth and the maximum number of students that can be registered into the module respectively.

The next Q lines will contain a single query each with the following format:

Query Type Input Format: `<QUERY_TYPE> <APPROPRIATE_PARAMETERS>`

1. `register STUDENT_NAME MODULE_CODE`
   Register the student with name STUDENT_NAME into module MODULE_CODE. The module is guaranteed to exist. If the student has already been registered into that module, print "STUDENT_NAME is already registered into MODULE_CODE". If the student is not registered but the module MODULE_CODE already has the maximum number of students, print "MODULE_CODE is full" and do not register the student. Otherwise, print "STUDENT_NAME successfully registered into MODULE_CODE ".

2. `remove STUDENT_NAME MODULE_CODE`
   Remove the student with name STUDENT_NAME from module MODULE_CODE. The module is guaranteed to exist. If the student is not registered into that module, print "STUDENT_NAME is not registered into MODULE_CODE". Otherwise, print "STUDENT_NAME has been removed from MODULE_CODE ".

3. `details STUDENT_NAME`

   List all of the modules that student STUDENT_NAME is registered for. If the student is not registered for any module, print "STUDENT_NAME has no modules". Otherwise, print the total number of MCs the student is taking, which is the sum of the MCs of all the modules the student is registered for. Then, list down in a single line the module codes of all the modules that student STUDENT_NAME is registered for and separate them with single spaces. Print the modules in alphabetical order. There is no whitespace at the end of the line.

4. `total`

   Print the total number of students currently registered into at least 1 module. A student registered into multiple modules is only counted once. If there are no students registered for modules, print "No students registered for modules".

5. `highest`

   Print, in a single line, the highest total number of MCs attained by any student, followed by a list of students who attained that number of MCs in alphabetical order. Separate these values using single spaces, and do not print a space after the last name. If there are no students registered for modules, print "No students registered for modules".

Output

Print the result of the query as described in the input format above. The last line of the output must contain a newline character. In the sample output below, the first few lines are left empty for better clarity of the sample. No blank lines are to be printed in the actual output.

Sample Input	Sample Output
4 17	
CS1010 4 2	
CS1020 4 2	
CS1101S 5 10	
CS1010R 1 5	
total	No students registered for modules
register John CS1020	John successfully registered into CS1020
register Kevin CS1020	Kevin successfully registered into CS1020
register William CS1020	CS1020 is full
details William	William has no modules
total	2
remove William CS1020	William is not registered into CS1020
remove Kevin CS1020	Kevin has been removed from CS1020
register William CS1020	William successfully registered into CS1020
register Kevin CS1010	Kevin successfully registered into CS1010
register Kevin CS1010R	Kevin successfully registered into CS1010R
register Kevin CS1010	Kevin is already registered into CS1010
details Kevin	5 CS1010 CS1010R
register Ivan CS1101S	Ivan successfully registered into CS1101S
remove John CS1020	John has been removed from CS1020
total	3
highest	5 Ivan Kevin

Explanation

There are 4 modules and a total of 17 queries. For the first query, no student has registered yet. In the next 2 queries, John and Kevin are registered into CS1020. After that, since CS1020 can only have a maximum of 2 students, William cannot be registered into CS1020 so "`CS1020 is full`" is printed. Thus, only John and Kevin have registered modules so the answer to the next query is 2.

Following that, removing William from CS1020 fails because he is not currently registered into CS1020. But after removing Kevin from CS1020, there is now a vacancy for William so he is successfully registered into CS1020. Then, Kevin registers into 2 other modules CS1010 and CS1010R for a total of 4 + 1 = 5 MCs. This is reflected when the details of Kevin are queried.

Afterwards, Ivan registers into CS1101S for a total of 5MCs as well. John is removed from CS1020 so this makes a total of 3 students registered, namely William, Kevin and Ivan. Since William only has 4 MCs, the highest total MCs by any student is 5 and Ivan and Kevin both have 5 MCs.

Hint

If you have an array list of strings with the name "students", you can sort the content of that array list lexicographically by calling "`Collections.sort(students);`". This method does not return a new array list but instead sorts the given array list.

Please refer to the Collections class on the Java API for more details.

Skeleton

You are given the skeleton file **IVLE.java.** You should see a file with the following content when you open it, otherwise you might be in the wrong directory.

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

import java.util.*;

public class IVLE {

    // define your own attributes, constructor, and methods here

    private void run() {

    }

    public static void main(String[] args) {
        IVLE ivle = new IVLE();
        ivle.run();
    }
}

class Module {
    // define your own attributes, constructor, and methods here
}

class Student {
    // define your own attributes, constructor, and methods here
}
```

Notes:

1. You should develop your program in the subdirectory `ex1` and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You only need to modify the skeleton file. You do not need to create a new file for each class. All code should be inside the file given to you in the `ex1` directory.
3. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm. You are free to define your own classes besides the ones given in the skeleton file.
4. You must (and need to) use OOP for this sit-in lab.
5. You are free to define your own methods.
6. Please be reminded that the marking scheme is:

   Input : 10%
   Output : 10%
   Correctness : 50%
   Programming Style : 30%, which consists of:
   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (incremental programming, proper modifiers [public / private]): 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%

   Compilation Error : Deduction of 50% of the total marks obtained.