Registration Data

The problem is not meant for analyzing the worst case time complexity of algorithms or data structures. It is more about using simple data structures as efficiently as possible and ensuring correctness of the program. There are more efficient data structures for some of these problems that we will do later, but for this lab you should use simple data structures that will work well for 'typical' usage.

You have to maintain a data structure to process registration data. You are given a list of courses being offered. Each course is assigned a slot and may have an upper bound on the number of students who can register for the course. You are also given the list of students on roll.

You have to process a sequence of operations. An operation may be to register a given student for a course, drop a course from a student's registration, print the students registered for a course, print the courses a student has registered for.

Some other operations are to find all common students between two courses and find common courses between two students.

There are some constraints that must be satisfied. A limit max_courses will be specified on the maximum number of courses a student can register for. A student cannot register for two courses in the same slot. The number of students in a course cannot exceed the specified limit.

Input Format. The first line of input will specify the number of courses, number of students, the maximum number of courses a student can register for, and the number of operations to be processed. The number of courses is at most 1000, number of students at most 100000, maximum number of courses a student can register for is at most 10, and the number of operations is at most 10^7.

The next lines will give the details of the courses. Each course is specified by a character string of length 6. This is followed by the slot number which is between 1 to 20 and the maximum number of registrations allowed for the course. This is -1 if there is no limit.

The next lines give the roll number of students, which are 9 character strings, one per line.

After that you are given a sequence of operations. Each operation will be specified in one of the following forms, one per line.

```
1. R rollno  course
   regsiter student rollno for course.
2. D rollno course
   drop course from student rollno's registration.
3. P rollno
   print the courses registered for by rollno.
4. P course
   print students registered for course
5. P rollno1 rollno2
```

```
   print common courses for students rollno1 and rollno2
6. P course1 course2
   print common students in courses course1 and course2.
```

The Output should be the outcome of each operation printed one per line, in the same order. If a registration operation is successful, print "success" else print "fail", without quotes. Note that the operation fails if either the student or the course is invalid, or the student is already registered for the course, or one of the constraints is violated. The same holds for drop, it fails if the student is not currently registered for the course. For all print operations, print the output in lexicographically sorted order on one line separated by a space. The operations must be performed in the given order, and the result is based on the registration at the time the operation is performed.

The amount of I/O is large so use scanf/printf OR set the flags for fast I/O with cin/cout. DO NOT use both.

You will get half credit if your programs work correctly for small inputs with say 100 courses, 10000 students and 10^5 operations. There is no fixed time limit, you should try to optimize the time that your program requires.

Submission. Submit a single file named RollNo_7.cpp.