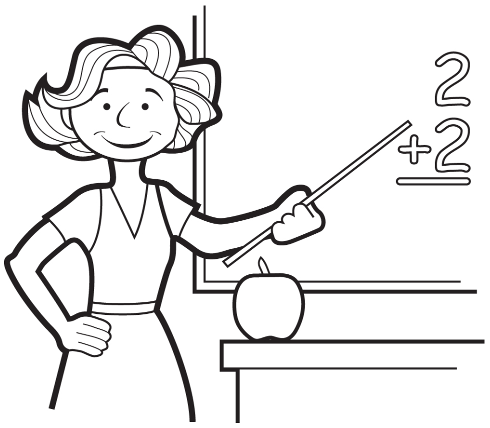
Class Scheduling



SIS is terrible! But the people in charge at UVa don't seem to notice (or, at the very least, are unwilling to admit it publicly). Instead, they want to add *more* features to it. But don't expect any increase in usability.

The new feature to be added to SIS is the ability to handle all the registrations at once, rather than have people register at various times. All the students will enter their requests, and the system will find the most optimal set of registrations (i.e. who gets into which course). Your job is to implement this algorithm.

For this problem, we designate a 'student registration request' which is one student signing up for one class. In general, students will sign up for multiple classes, thus causing multiple 'student registration requests' for the same student.

The algorithm desired will try to schedule all the students into some set of the classes that they desire. The algorithm is subject to a few constraints:

- An optimal set of registrations may cause an individual student to be enrolled in too few or too many classes. The algorithm must ensure that this does not happen (although that may not be possible, given the constraints). For simplicity's sake, we'll just have a number n which is the *exact* number of courses each student must be enrolled in: no more, no less.
- Each course has a course cap which is the maximum number of students that can enroll in the course.

Note that the number of courses that a student should be enrolled in is provided only once, and applies to all students. The course cap is course-specific, and thus is provided once for every course.

Input

The input will consist of multiple cases. The first line of each input case will contain three integers, r, c, and n: r is the number of student registration requests, c is the number of courses, and n is the number of classes each student is to be enrolled in. Note that n applies to all students.

The following r lines will contain two strings each: the name of a student followed by the name of a class s/he wants to register for. Both the names of the students and the names of the courses will contain only alphanumeric characters and underscores – no white space or other punctuation (other than an underscore) will be allowed in the student or course names. Student and course names will not start with an underscore.

The following c lines will contain a the list of courses and the maximal capacity for that course. The format for the course names is as defined in the preceding paragraph. The maximum capacity for each course will be listed on the same line, space separated, as an integer. You will not see a course listed here that was not listed in one of the r lines above; likewise, all the courses listed in the r lines above will be listed here.

Each input case will be followed by a blank line.

Three zeros (for r, c, and n) will signify the end of the input.

Output

For each test case, the output will be either 'Yes' or 'No', depending on whether it is possible to schedule the students in n of their desired courses, given the constraints and the registration requests. Presumably, your algorithm will have figured out *what* that matching is; but for this output, only 'Yes' or 'No' is to be output.

Specifically, if there are s students, then there should be $s * n$ fulfilled registration requests (and each student should have exactly n fulfilled registration requests).

The maximum number of course registration requests will fit in a 32-bit integer.

Sample Input

```
9 3 2
Alice CS_2102
Alice CS_3102
Alice CS_4102
Bob CS_2102
Bob CS_3102
Charlie CS_2102
Charlie CS_4102
David CS_2102
David CS_3102
CS_2102 3
CS_3102 3
CS_4102 3

0 0 0
```

Sample Output

```
Yes
```

Input Case Diagram

The input case shown above corresponds to the following graph. Each student is listed on the left, and is abbreviated by the first letter of his or her name. Each course is listed on the right, and is abbreviated by the first digit of the course number.