Problem U

The nycsubmit System

Problem ID: nycsubmit

Your solution to this problem must be implemented using an object-oriented approach. Any other solution will receive zero points, even if the testing system judges it correctly.

The University of Chicago's Department of Computer Science created their own system for managing the submission and grading of programming assignments called chisubmit. Based on the success of chisubmit, a university in New York City is implementing their own version, called nycsubmit. Like us, they have decided to model the various aspects of a university course (assignments, submissions, instructors, graders, etc.) using classes and objects. For this problem, we will focus on defining classes for an assignment and a course.

An assignment is defined to have at least three components: S, N, and T. S is a unique identifying string, N is a descriptive assignment name, and T is the total number of points for the assignment. For example, the first programming assignment within a computer science course could have `S="pa1"`, `N="Assignment #1"` and `T=100`. Each assignment must maintain a list of students who submitted a solution for the assignment. The list contains each student's unique identifier (U) as an integer (e.g., `U=10123456`). For this problem, you can assume that students submit their actual solution via some other component of the system and an assignment just holds the identifiers of the students who submitted a solution.

Every course includes a unique course code (S), a descriptive course name (N), and the total number of students in the course (P). Each course will also manage the assignments within the course. For example, a course could have a course code of `S="CMSC 14100"`, a descriptive name of `N="Introduction to Computer Science I"`, and a total number of students equal to 60 (`P=60`).

Your task for this problem is to design and implement two classes: `Assignment`, and `Course`. You are responsible for determining the appropriate attributes (also known as instance variables) and methods for each class. The only required methods are `total_points` and `percent_submitted` inside the `Course` class:

- Implement a method called `total_points` in the `Course` class that computes the total number of points for all assignments in the course.
- Implement a method called `percent_submitted` in the `Course` class the computes the percent of students who have submitted a given assignment. This method takes in one argument, `aid`, that represents an assignment identifier S. This method returns a float between 0.0 and 100.0 with the percent of students in the course who have made a submission for that assignment. An appropriate value of your choosing can be returned if there is no assignment with the identifier `aid`. For the purposes of this problem, the number of students who have submitted the assignment is just the number of submissions in the `Assignment` class (i.e., you don't need to validate whether a student is registered in the class, etc).

As a reminder, your solution to this problem must be implemented using an object-oriented approach. Any other solution will receive zero points, even if the testing system judges it correctly.

Input

The input contains information pertaining to assignments, and submissions for each assignment for a single course.

The input begins with a line containing information about the course. A course line contains S, N, and P ($1 \leq P \leq 200$) as described above. S and N contain only lowercase letters, digits and underscores, no whitespace, and has a maximum length of 50 characters. Each line component is separated by a single space.

The second input line contains two integers X ($1 \leq X \leq 20$) and Y ($1 \leq Y \leq P * X$), each separated by a single space. X is the total number of assignments in the course and Y is the total number of submissions for all assignments in the course.

The third input line contains W number of assignment identifiers; each separated by a single space. You will use these assignment identifiers to compute the percent of students who have submitted a solution for the assignment.

The input is followed by X lines, each corresponding to a single assignment. A assignment line contains S, N, and T ($1 \leq T \leq 100$) as described above. S and N contain only lowercase letters, digits and underscores, no whitespace, and has a maximum length of 50 characters. Each line component is separated by a single space.

The input is followed by Y lines, each corresponding to a single submission for an assignment. A submission line contains U and S. U is an 8 digit integer that represents the unique identifier for a student. S the assignment identifier for the submission. Each line component is separated by a single space. You can assume that each student identifier will only make a single submission for a specific assignment and there will be at most P number of submissions for an assignment.

Output

The output is the total number of points for all assignments within the course and the percent of students who have submitted a solution for the assignment identifiers indicated on the third line of the input.

The first line of the output is total number of points for all assignments within the course.

The output is followed by W lines, each corresponding the percent of students who have submitted a solution for the assignment identifier on the third input line. Each line contains the assignment identifier S, followed by the percentage as floating point value. If the identifier cannot be found then it prints `"INVALID"` for that specific assignment identifier; otherwise it prints the percent of students who have submitted a solution. Each line component is separated by a single space. The order of the lines must match the order of the W identifiers on the third input line.

Sample Input 1

```
CMSC_14100 Intro_to_Computer_Science_1 4
3 5
pa2 pa1
pa1 Programming_Assignment_1 50
pa2 Programming_Assignment_2 100
pa3 Programming_Assignment_3 100
10000001 pa1
20000002 pa1
30000003 pa1
10000001 pa2
20000002 pa2
```

Sample Output 1

```
250
pa2 50.0
pa1 75.0
```

Sample Input 2

```
CMSC_14200 Intro_to_Computer_Science_2 4
3 5
pa2 pa1 pa3
pa1 Programming_Assignment_1 25
pa2 Programming_Assignment_2 25
pa3 Programming_Assignment_3 50
10000001 pa1
20000002 pa1
30000003 pa3
10000001 pa2
20000002 pa2
```

Sample Output 2

```
100
pa2 50.0
pa1 50.0
pa3 25.0
```

Sample Input 3

```
CMSC_14300 Systems_Programming_I 2
2 2
pa2 pa4 pa1 pa3
pa1 Programming_Assignment_1 25
pa2 Programming_Assignment_2 25
10000001 pa1
20000002 pa1
```

Sample Output 3

```
50
pa2 0.0
pa4 INVALID
pa1 100.0
pa3 INVALID
```