Description

You have done a great job! As a reminder, our online learning platform offers its students four courses: Python, DSA, Databases, and Flask. To complete each of these courses, a student must earn a certain number of points that are different for each course: 600 for Python, 400 for DSA, 480 for Databases, and 550 for Flask. Students can enroll in any of the courses and take as many assignments as they want.

It would be a good idea to know which course is the most popular, which is the hardest to beat, and so on. For this purpose, you need to add new features to your program: it must provide the statistics about each course and track the performance of each student in each course.

Calculate the following:

- Find out which courses are the most and least popular ones. The most popular has the biggest number of enrolled students;
- Find out which course has the highest and lowest student activity. Higher student activity means a bigger number of completed tasks;
- Establish the easiest and hardest course. The easiest course has the highest average grade per assignment;
- Establish top learners for each course.

A student is enrolled in the course if the learning progress data contains at least one non-zero submission for that course. See the following example:

```
182365 4 0 0 8
182365 0 0 0 5
182366 0 8 0 4
```

On the first line, student `182365` completed one Python task and earned 4 points and also completed one Flask task and earned 8 point. On the second line the same student completed one Flask task and was awarded 5 points. On the third line student `182366` completed one DSA task and received 8 points and one Flask task and got 4 points. In this example, the student `182365` is considered to be enrolled in the Python and Flask courses, and the student `182366` is considered to be enrolled in the DSA and Flask courses.

One important thing. During statistics calculation, if multiple courses qualify for any of these categories, list the names of all such courses. If any course is already included in a category, it cannot be included in the opposite category. For example, if the Python Course is listed in the Highest Activity category, it cannot be listed in the Lowest Activity category.

Look at the example above. The most popular course is Flask (2 enrolled students), the least popular course is Databases (0 enrolled students), the highest activity is Flask (3 submissions), the lowest activity is Databases (0 submissions), the easiest course is DSA (average score is 8), and the hardest course is Python (average score is 4):

```
Type the name of a course to see details or 'back' to quit:
Most popular: Flask
Least popular: Databases
Highest activity: Flask
Lowest activity: Databases
Easiest course: DSA
Hardest course: Python
```

If no course falls into a certain category, for example if no students have enrolled in any of the courses or data can't be retrieved, print `n/a`. See more examples in the Examples section below.

Information about top learners should be presented as a list containing the following information: a student's ID, the total points for a course, and the course completion progress as a percentage:

```
Python
id     points completed
125684 423    70.5%
200751 420    70.0%
130400 405    67.5%
```

The list must be sorted by the total number of points in descending order, and if two or more students have the same number of points, they must be sorted by their ID in ascending order. Use the dot `.` character as the decimal point and `round()` function for rounding decimal numbers.

Also, use the following notation for course names: `Python`, `DSA`, `Databases`, `Flask`.

This stage has a lot of requirements, so you can use unit tests to break things down.

Objectives

In addition to the features of the previous stages, your program should:

1. Add a new command to your program's toolkit: `statistics`. If users enter this command, your program should output the header: `Type the name of a course to see details or 'back' to quit` and six lines with the following information: `Most popular`, `Least popular`, `Highest activity`, `Lowest activity`, `Easiest course`, `Hardest course` with the names of the corresponding courses. After that, if users enter a course name, the program should display the details of this course, but if users enter a name that doesn't correspond to any of the courses, the program should print `Unknown course`. When the `back` command is entered, the program goes back to other available commands.

2. When users enter the `statistics` command, your program must display the details about any course. When users type in the name of a course, the program should display the name of the course in the first line, then the column headers, and a list of student IDs, their total points in the respective course, and the percent of completion (one decimal place precision). If a course has no students, output only the name of the course and the column headers.

3. Course details are available only after users enter `statistics`, they should not be available with the `back`. If users type in any course name before entering the `statistics` command, the program must respond with the `Unknown command!` message.

4. Sort student lists by the total number of points in descending order and then by the ID in ascending order.

Examples

The greater-than symbol followed by a space (`> `) represents the user input. Note that it's not part of the input.

Example 1: *showing statistics with no data available*

```
Learning Progress Tracker
> statistics
Type the name of a course to see details or 'back' to quit:
Most popular: n/a
Least popular: n/a
Highest activity: n/a
Lowest activity: n/a
```

```
Easiest course: n/a
Hardest course: n/a
> python
Python
id      points  completed
> swing
Unknown course.
> back
> exit
Bye!
```

Example 2: *showing statistics*

```
Learning Progress Tracker
> add students
Enter student credentials or 'back' to return:
> John Doe johnd@email.net
The student has been added.
> Jane Spark jspark@yahoo.com
The student has been added.
> back
Total 2 students have been added.
> list
Students:
10000
10001
> add points
Enter an id and points or 'back' to return:
> 10000 8 7 7 5
Points updated.
> 10000 7 6 9 7
Points updated.
> 10000 6 5 5 0
Points updated.
> 10001 8 0 8 6
Points updated.
> 10001 7 0 0 0
Points updated.
> 10001 9 0 0 5
Points updated.
> back
> statistics
Type the name of a course to see details or 'back' to quit:
Most popular: Python, Databases, Flask
Least popular: DSA
Highest activity: Python
Lowest activity: DSA
Easiest course: Python
Hardest course: Flask
> python
Python
id    points    completed
10001 24        4.0%
```

```
10000 21        3.5%
> dsa
DSA
id    points    completed
10000 18        4.5%
> databases
Databases
id    points    completed
10000 21        4.4%
10001 8         1.7%
> flask
Flask
id    points    completed
10000 12        2.2%
10001 11        2.0%
> back
> exit
Bye!
```