CS200: Programming I **Instructor:** Sam Vinitsky

P11: CS200 Final Grades

(40 points)

Due: Thursday, August 8 @ **1:00pm**

Part 0: Introduction

Brrrring brriiiiiing the phone rings – it's me, Sam, local computer science instructor! After watching you help person after person throughout the course of the semester, I have decided it's finally time for you to help **me**. It's the end of the semester, and I'm very busy with important administrative tasks such as writing lectures, grading homeworks, and coming up with bad puns for the final exam. However, I still need to determine your finals grades for the course... So I've decided to outsource that task to you![1]

You will be writing a program called `FinalGrades.java` to compute the final grades for everybody in the course. First, you will write a function to compute everyone's raw scores,[2] then you will figure out how to assign final letter grades so that a certain number of people receive each letter.[3]

Additionally, **you will be writing a grading script to grade your own assignment**.

Since I will be using the programs you turn in to figure out final grades, I will need them by **August 8st, at 1:00pm**. By university policy, **I cannot accept any assignments after this time**. This means if your assignment is one minute late, I cannot accept it. Additionally, this means that **no slip days may be used for this assignment**.

Each program you turn in must include a comment at the top with (1) your full name, (2) your student ID number, (3) your netID, and (4) the name of anyone you discussed the homework with (excluding Sam and Alex).

Before attempting this homework, make sure you work through the practice problems from this week's lectures (including those on ArrayLists!).

As always: **start early, ask questions, and have fun!**

[1] Note: You will not actually be allowed to compute your own final grade, or the final grades for anyone else in the course. This is merely an academic exercise.

[2] **Disclaimer:** this write-up contains no information about real students. All names and grades given as examples here are made up.

[3] This is not exactly how I will be assigning final letter grades, but it's close enough.

Part 1: Calculating final grades [17 points]

As stated in the introduction, I am a very busy person at this time of the semester. I need you to help me calculate the final grades for everyone in the course. Write a program called `FinalGrades.java`. You will write some functions, but you do not need to include anything in `main` (just use this to test your functions!).

a) Calculating raw score [7 points]

First, you will need to calculate each student's raw score, which is a double between 0 and 1 representing their "percentage" (i.e. `0.93` would represent a 93% in the course). Each student's raw score is the proportion of total possible points they earned over the course of the semester. For example, if Bobby earned 100 points out of a possible 200 points, his raw final score would be $100/200 = 0.5$. If Penny earned 10 out of 15 possible points, her final score would be 0.666666.

Write function called `computeRawFinalGrades` to compute the raw score for each student in the course. Your function will use the following data structures:

- `grades`
  - a 2-dimensional array of `double`'s
  - `grades[i][j]` represents the number of points that student number `i` earned on assignment number `j`
- `maxPoints`
  - a 1-dimensional array of `int`'s
  - `maxPoints[j]` represents the maximum number of points possible on assignment `j`
- `rawGrades`
  - a 1-dimensional ArrayList of `Double`'s
  - The number at spot `i` of this array is the raw final grade of student number `i`. This is between 0 and 1.

`computeRawFinalGrades` should take as input `grades` and `maxPoints`, and should return `rawGrades`. The signature of your function must be exactly:

```
public static ArrayList<Double> computeRawFinalGrades(double[][] grades, int[] maxPoints)
```

If the inputs are invalid for any reason (i.e. the lengths don't match up correctly, or the arrays are empty), then your program can crash. That's okay.

For examples of what your function should output, see the test cases listed in part 2.

b) Computing Letter Grades [20 points]

At the end of the semester, I need to assign each of you a "letter grade" based on how well you have performed over the course of the semester. Again, I am very busy, so I have decided to let you help me!

I have decided to give out a certain number of each letter grade, but I have not yet decided HOW many will get each grade. For example, maybe 20 students will get A's, 5 will get AB's, 20 will get B's, and 10 will get C's. Or maybe 10 will get A's, 30 will get B's, and there will be no C's. Since this is *not fixed*, this will need to be given as input to your function.

Write a function called `computeLetterGrades` to help me compute letter grades for each student. Your program will use the following data structures:

- `rawGrades`
  - a 1-dimensional ArrayList of `Double`'s
  - The number at spot `i` of this array is the raw final grade of student number `i`. This is between 0 and 1.
- `letters`
  - a 1-dimensional array of `String`'s
  - `letters[i]` is a letter grade – i.e. "A" or "B" or "BC", etc. The BEST grade comes first, and then the second best, etc... the worst grade is last.
- `breakdown`
  - a 1-dimensional array of `int`'s
  - For each `i < letters.length()`, `breakdown[i]` represents the number of students who should get `letters[i]`
  - The length of `breakdown` is ONE LESS than the length of `letters`. Every student not accounted for in `breakdown` gets the last grade in `letters`.
  - Ex: If `letters` is `{"A", "B", "C", "D"}`, and `breakdown` is `{10, 5, 15}`, then the students with the 10 best scores get an "A", the next 5 best get a "B", the next 15 best get a "C", and the rest get "D"s.
  - Ex: If `letters` is `{"QQ", "A", "X"}`, and `breakdown` is `{2, 3}`, then the 2 best students get best scores get a "QQ", the next 3 best get an "A", and the rest get "X"s.

- `finalLetterGrades`
  - a 1-dimensional array of `String`'s
  - `finalLetterGrade[i]` is the final letter grade of student number `i`

`computeLetterGrades` should take `rawGrades`, `breakdown` and `letter` as inputs, and should return `finalLetterGrades`. The signature of your function should be **exactly**:[4]

```
public static String[] computeLetterGrades(ArrayList<Double> rawGrades,
                                String[] letters, int[] breakdown)
```

If the inputs are invalid for any reason (i.e. the lengths don't match up correctly, or the arrays are empty), then your program can crash. That's okay.

For examples of what your function should output, see the test cases listed in part 2.

Part 2: Grading `FinalGrades.java` [10 points]

Now that you've completed `FinalGrades.java`, I need to actually *grade* your programs... However, as I've mentioned repeatedly, I am very busy, so I will let you do that for me!

Write a program called `GradeP11.java` that will grade this assignment for me. When you run `GradeP11.java`, it should run each function in `FinalGrades.java` for a variety of inputs, and confirm that they produce the correct outputs. Then, you will output the number of tests cases that passed.

The inputs you must test are listed in the following section. I recommend that each test follow this structure:

1. Create a variable for each input
2. Create a variable, and give it the expected output
3. Call the function[5] with the proper input variables, storing the return value in a variable
4. Compare the return value with the expected output[6]

[4] I had to split mine over two lines to get it to fit in the PDF nicely, but this doesn't change the *signature* – i.e. the function name, and input/output types/numbers.

[5] If you put `FinalGrades.java` and `GradeP11.java` in the same folder, you should be able to do this by writing `FinalGrades.functionName()` in your code (where `functionName` is the name of the function you want to call

[6] if these are arrays or ArrayLists, you will need to check that *each* element is correct, since we cannot compare them directty using `==` or `.equals()`...

5. If they are the same, then this test passed

6. Otherwise, this test failed

When your program passes a test, it should print out "FunctionName: Test #X passed!" on its own line (where "X" is the test number, and FunctionName is the name of the function you're testing). When your program fails a test, it should print out: "FunctionName: Test #X failed!". After running all of the tests for a function, your program should print out "FunctionName: Total number passed: A/B" where A is the number of tests passed, and B is the total number of tests.

Remember that there are **two** functions you must check. Their test cases are listed separately below. Your program should test BOTH of them.

a) Test cases for `computeRawFinalGrades`:

Here is the list of all the test cases your program must check for `computeRawFinalGrades`. **Additionally, you must add 4 cases of your own, ONE of which has more then 10 students, and ONE of which more than 8 assignments. These can be different ones. The other two test cases can be anything.**

Remember that this function will return an ArrayList, not an array!

Note: since you are comparing *doubles*, you will run into the issue where your answer is off by 0.00001 or less. This is because of how doubles are stored in Java. In order to accommodate for this, you should check that your answers come within 0.00001 of the expected answer.

grades	maxPoints	return
{5,6,7}, {0,0,0}, {10,12,14}	{10,12,14}	[0.5, 0.0, 1.0]
{1,2,3,4}, {2,4,6,8}	{3,6,9,12}	[0.3333333333, 0.6666666666]
{0,10}, {5,5}, {7,3}	{10,10}	[0.5, 0.5, 0.5]
{5,1}, {10,20}, {30,30}, {11,10}	{30,30}	[0.1, 0.5, 1.0, 0.35]
{4,10,18,19,15,39,21}	{5,10,20,20,20,40,30}	[0.86896551724]

b) Test cases for `computeLetterGrades`:

Here is the list of all the test cases your program must check for `computeLetterGrades`. **Additionally, you must add 4 cases of your own, each of which has more then 10 students, and more than 5 grade letters.**

rawGrades	letters	breakdown	return
{0.5, 0.0, 1.0}	{"A","B", "C", "F"}	{1, 1, 1}	{"B", "C", "A"}
{0.95, 0.99, 0.55, 0.98}	{"A", "AB", "B","D", "F"}	{2,1,0,1}	{"AB", "A", "D", "A"}
{0.95, 0.99, 0.55, 0.98}	{"A", "AB", "B","D", "F"}	{2,1,0,0}	{"AB", "A", "F", "A"}
{0.1, 0.2, 0.3, 0.15}	{"X", "Q", "LL"}	{1,2}	{"LL", "Q", "X", "Q"}
{0.1, 0.2, 0.3, 0.15}	{"X", "Q", "LL"}	{4,0}	{"X", "X", "X", "X"}
{0.1, 0.2, 0.3, 0.15}	{"X", "Q", "LL"}	{0,4}	{"Q", "Q", "Q", "Q"}
{0.1, 0.2, 0.3, 0.15}	{"X", "Q", "LL"}	{0,0}	{"LL", "LL", "LL", "LL"}

Part ∞: Feedback Form [3 points]

Fill out this feedback form **after you submit** this assignment. Completion of this will count towards your grade, but your responses themselves will not affect your grade in any way (so be honest!).

What to turn in

On Canvas, turn in a zip folder called `<my_net_id>_P11.zip` with the following programs:

- `FinalGrades.java` [17 points]
- `GradeP11.java` [20 points]

Also complete the feedback form. [3 points]
