Complex Engineering Activity

Two groups of students at a local university are enrolled in certain special courses during the summer semester. The courses are offered for the first time and are taught by different teachers. At the end of the semester, both groups are given the same tests for the same courses, and their scores are recorded in separate files. The data in each file is in the following form.

courseId1 score1 score2, . . . scoreN -999
courseId2 score1 score2, . . . scoreN -999
.
.
.
Let us write a program that finds the average course score for each course for each group. The output is of the following form:

CourseId	Group No	Course Average
CSC	1	83.71
	2	80.82
ENG	1	82.00
	2	78.20

.
.
.
Avg for group 1: 82.04
Avg for group 2: 82.01

Input: Because the data for the two groups are recorded in separate files, the input appears in two separate files.

Output: As shown above.

Reading input data from both files is straightforward. Suppose the data is stored in the file **group1.txt** for group 1 and file **group2.txt** for group 2. After processing the data for one group, we can process the data for the second group for the same course and continue until we run out of data. Processing data for each course is similar and is a two-step process.

1. a. Sum the scores for the course.
   b. Count the number of students in the course.
   c. Divide the total score by the number of students to find the course average.

2. Output the results.

We are comparing only the averages of the corresponding courses in each group, and the data in each file is ordered according to course ID. To ensure that only the averages of the corresponding courses are compared, we compare the course IDs for each group. If the corresponding course IDs are not the same, we output an error message and terminate the program.

The discussion suggests that we should write a function, **calculateAverage**, to find the course average. We should also write another function, **printResult**, to output the data in the form given. By passing the appropriate parameters we can use the same functions, **calculateAverage** and **printResult**, to process each course's data for both groups.

The preceding discussion translates into the following algorithm.

1. Initialize the variables.
2. Get the course IDs for group 1 and group 2.
3. If the course IDs are different, print an error message and exit the program.
4. Calculate the course averages for group 1 and group 2.
5. Print the results in the form given above.
6. Repeat steps 2 through 5 for each course.
7. Print the final results.

You have to write a code using the concept of file handling for this problem. You also have to submit a report in which you explain your code step by step in detail with code snippets.