Project 3: CSC 24400
Student Records Again

Description:

For your third programming assignment, you are to write a program to open and read input files which contain an unknown number of records and process the student data in accordance with the instructions given below. Your program is to be modularized with functions to sort, print, and read. It is also to include any other functions you deem necessary.

INPUT:

The input file for this program consists of an unknown number of records (You may assume a maximum of 100 records in the data file). Each record will consist of three lines of data. The first line will contain a Student's ID Number, which will be an integer. The second line will contain a first name and a last name separated by at least one whitespace character. The third line will contain **four (4)** test scores of type float. A negative value for the Student's ID Number will act as the sentinel, indicating the end of input. If there are entries after the sentinel, they should be ignored by your program. The input file name will be **data3.txt**.

PROCESSING:

You will need to create an array with 100 elements to hold the student IDs. You will also need a multidimensional array for this project to hold the student's name (first name and last name). This array should have 100 rows and 2 columns, where the first column will hold the first name and the second column will hold the last name. This array needs to be of type "string *" so that each column can point to a c++ string to hold the names (these strings will be allocated on the heap using dynamic memory allocation). You will need a single dimensional array for the test scores. This array should have 100 elements, and should be of type "float *" to hold the test scores. This third array will point to arrays of type float where the first four columns, columns 0-3, hold the test scores and the last column, column 4, holds the average grade (each array for an individual student's test scores and test average will be allocated on the heap, as separate arrays, using dynamic memory allocation). Each of these arrays for the test scores will be allocated in dynamic memory and should be of size 5. Finally, you will need an array of char, which will hold the student's letter grades. You should initialize all entries in the student IDs array to -1, initialize all entries in the student name array to NULL, initialize all entries in the scores array to NULL, and initialize all characaters in the letter grades array to the space character. Hence, your arrays should be declared something like:

```
int     IDs[100];
string *name[100][2];
float  *scores[100];
char    letter[100];
```

Here, the same row in each of the arrays is for the same student. Column 0 of the name array will hold the student's first name and column 1 of the name array will hold the student's last name. You may assume that a name does not have any whitespace characters in it.

The processing steps are:

1) Read all the data from the input file into the appropriate arrays. As you read in the names, you will need to allocate two strings in dynamic memory to hold the two names. If variable **`row`** contains the

row index for the student, then the `name[row][0]` element should point to the string allocated to hold the first name and the `name[row][1]` element should point to the string allocated to hold the last name. As you read in the test scores you will need to allocate an array of size 5 in dynamic memory to hold the 4 test scores and the average. Be sure to make the `scores[row]` element point to this array on the heap. When you allocate each of these arrays for the test scores, you should initialize it to contain a value of -1 in each of its 5 elements before proceeding to store the test scores.

2) **After** reading in **all** the data, the program is to print the list of student IDs, student names, and all their test scores (in other words, print out the contents of the arrays), but not the average scores or the letter grades.
3) After this information is printed, the program is to sort the listing of students (and their names and test scores), by student ID number from high to low, and print the list again.
4) Then the program is to sort the listing of students in alphabetical order from A to Z by Last Name and print the contents of the arrays.
5) The program is then to calculate the student's test average and letter grade for each student, and store that average into column 5 of the array pointed to by the scores array for that student, and store the letter grade in the letter array element for that student. The course grade is to be one of the five standard letter grades of A, B, C, D, or F.
6) The program should then print the student information (ID, test scores, average, and letter grade) for each student.
7) Using the test score average as the "key", the program is then to sort the student information in the order of highest test average to lowest test average.
8) The program is then to print a final list of the students, their test scores, averages, and letter grade.
9) After all of the above steps are finished, you must go back and <u>delete</u> all of the dynamically allocated memory in the name array and the scores array.

   Note: You should have two print functions. The first will print the student IDs, student names, and the test scores. The second print function will print all of the contents of all four arrays.

OUTPUT:

All output from this program must be generated to the screen/terminal and to the output file "output.txt." All output in the file must be identical to the output generated to the screen, and vice versa.

The output for this program is to consist of a listing of the student information according to the sample output. Output may begin in any column. All output must be properly labeled and be in an easy to read format. All test scores are to have 1 decimal point accuracy, while the <u>average of the test scores</u> is to have 2 decimal place accuracy. Course grades are to be uppercase letters. *<u>Output for a single student's listing MUST NOT be split between two pages.</u>* A message such as **END OF PROGRAM OUTPUT** must appear after all information has been printed. All output must be stored in the file **output2.txt.**

SAMPLE Input: Suppose **data3.txt** looked like this:

```
5601
Charles Babbage
66.7 83.4 88.8 72.1
8202
John    VonNeumann
```

```
44.0 90.5 83.2 79.2
3303
Ada   Lovelace
93.5 72.6 80.9 96.5
4104
Blase Pascal
99.6 88.6 94.6 92.7
9905
Herman Hollerith
62.3 70.2 55.5 50.3
1206
Grace Hopper
80.9 90.7 85.1 87.4
7307
Alan Turing
70.2 70.4 75.7 78.3
-999
```

Then your output should look something like this (both on the screen and in the output file):

```
Michelle Lynn CSC 24400 Section 11
Fall 2017 Assignment #2
----------------------------------------------------------------------

The original student data is:
Student ID    FirstName   Last Name   Test 1  Test 2  Test 3  Test 4
------------  ----------  ----------  ------  ------  ------  ------
    5601      Charles     Babbage      66.7    83.4    88.8    72.1
    8202      John        VonNeumann   44.0    90.5    83.2    79.2
    3303      Ada         Lovelace     93.5    72.6    80.9    96.5
    4101      Blaise      Pascal       99.6    88.6    94.6    92.7
    9905      Herman      Hollerith    62.3    70.2    55.5    50.3
    1206      Grace       Hopper       80.9    90.7    85.1    87.4
    7307      Alan        Turing       70.2    70.4    75.5    78.3

The list of students sorted by ID Number high to low is:
Student ID    FirstName   Last Name   Test 1  Test 2  Test 3  Test 4
------------  ----------  ----------  ------  ------  ------  ------
    9905      Herman      Hollerith    62.3    70.2    55.5    50.3
    8202      John        VonNeumann   44.0    90.5    83.2    79.2
    7307      Alan        Turing       70.2    70.4    75.5    78.3
    5601      Charles     Babbage      66.7    83.4    88.8    72.1
    4101      Blaise      Pascal       99.6    88.6    94.6    92.7
    3303      Ada         Lovelace     93.5    72.6    80.9    96.5
    1206      Grace       Hopper       80.9    90.7    85.1    87.4

The list of students sorted by Last Name from A to Z:
Student ID    FirstName   Last Name   Test 1  Test 2  Test 3  Test 4
------------  ----------  ----------  ------  ------  ------  ------
    5601      Charles     Babbage      66.7    83.4    88.8    72.1
    9905      Herman      Hollerith    62.3    70.2    55.5    50.3
    1206      Grace       Hopper       80.9    90.7    85.1    87.4
    3303      Ada         Lovelace     93.5    72.6    80.9    96.5
    4101      Blaise      Pascal       99.6    88.6    94.6    92.7
    7307      Alan        Turing       70.2    70.4    75.5    78.3
    8202      John        VonNeumann   44.0    90.5    83.2    79.2

The list of students with their test average and course grade is:
                                                                Test     Course
Student ID    First Name  Last Name   Test 1  Test 2  Test 3  Test 4  Average  Grade
------------  ----------  ----------  ------  ------  ------  ------  -------  -----
    5601      Charles     Babbage      66.7    83.4    88.8    72.1    77.75     C
    9905      Herman      Hollerith    62.3    70.2    55.5    50.3    59.58     F
    1206      Grace       Hopper       80.9    90.7    85.1    87.4    86.03     B
    3303      Ada         Lovelace     93.5    72.6    80.9    96.5    85.88     B
    4101      Blaise      Pascal       99.6    88.6    94.6    92.7    93.88     A
    7307      Alan        Turing       70.2    70.4    75.5    78.3    73.60     C
    8202      John        VonNeumann   44.0    90.5    83.2    79.2    74.23     C


The list of students sorted by test average high to low is:
                                                                Test     Course
Student ID    First Name  Last Name   Test 1  Test 2  Test 3  Test 4  Average  Grade
------------  ----------  ----------  ------  ------  ------  ------  -------  -----
    4101      Blaise      Pascal       99.6    88.6    94.6    92.7    93.88     A
    1206      Grace       Hopper       80.9    90.7    85.1    87.4    86.03     B
    3303      Ada         Lovelace     93.5    72.6    80.9    96.5    85.88     B
    5601      Charles     Babbage      66.7    83.4    88.8    72.1    77.75     C
    8202      John        VonNeumann   44.0    90.5    83.2    79.2    74.23     C
    7307      Alan        Turing       70.2    70.4    75.5    78.3    73.60     C
    9905      Herman      Hollerith    62.3    70.2    55.5    50.3    59.58     F

 --------------------------------
|      END OF OUTPUT             |
 --------------------------------
```

Don’t forget to print the header and the footer!

Project Submission:

Create a .zip file containing your .cpp and .h files ONLY (do not include executables or object files please – these take large amounts of disk space). Submit your .zip file to the project submission tool on the Canvas course web page for this assignment. **All submissions must be received by 11:59 p.m. the day they are due in order to receive full credit**.