Project #03: Grade Analysis console app

Complete By: Wednesday February 13th @ 11:59pm
Assignment: Build a console-based Grade Analysis app
Policy: Individual work only, late work *is* accepted (up to 48 hours late for 10% penalty)
Submission: via GradeScope

Assignment

Here in project #03 we're going to put your GradeUtil functions to work as part of a Grade Analysis console application. The input file will be grade data for a particular college and semester. The application will input the data, parse, and then accept commands from the user to search and analyze this data. In particular, the application will support the following 5 commands:

1. Summary of a department or college
2. Search by course # or instructor prefix
3. List of courses with grading type of unknown
4. List of courses by DFW rate
5. List of courses by percentage of A letter grades

Input Files

All input files are CSV files that contains two header rows followed by 1 or more rows of data. Here's the start of the "fall-2018.csv" file that is provided on the course web page:

```
Engineering,Fall,2018
Dept,Number,Section,Title,A,B,C,D,F,I,NR,S,U,W,Instructor
CS,377,1,Communication and Ethics,93,48,2,0,0,0,0,0,0,3,Burton
ME,308,1,Mechanical Vibrations,16,26,21,9,3,0,0,0,0,0,Szwalek
ME,599,1,PhD Thesis Research,0,0,0,0,0,0,11,0,0,0,Shahbazian-Yassar
ME,211,2,Fluid Mechanics l,3,9,8,9,3,1,0,0,0,0,Mirbod
ECE,559,1,Neural Networks,29,46,2,0,0,0,0,0,0,0,Koyuncu
.
.
.
CME,211,1,Fluid Mechanics and Hydraulics,20,21,4,3,0,0,0,0,0,1,Vitousek
```

The first line consists of three values: college name, semester, and year. The second line consists of the column headers, and can be ignored. The remaining lines are data rows, each with 15 values. "I" refers to the

# of incompletes, "W" the # of withdrawals, and "NR" the # of "no reports" --- e.g. students who never attended class and thus no grade could be reported. More commonly, grades of NR occur when grades are not submitted before the end of semester deadline; in this case all grades for the course are in the form of NRs. **Note**: the format of a data row matches the format of the string accepted by the **ParseCourse()** function.

Assignment Details

The program starts by inputting the name of the input file from the keyboard, parsing this data, and then outputting a college-level summary of this data. For example, here's what we get for "fall-2018.csv":

```
fall-2018.csv
** College of Engineering, Fall 2018 **
# of courses taught: 247
# of students taught: 13715
grade distribution (A-F): 44.62%, 31.28%, 14.89%, 5.33%, 3.89%
DFW rate: 12.78%

Enter a command>
```

Note that we will be auto-grading using Gradescope, so your output must match what you see *exactly*. Any space you see will be exactly 1 blank character, and that's a blank line between "DFW rate" and "Enter". All real numbers are output with 2 digits to the right of the decimal point. This is obtained by #include <iomanip> and executing these 2 lines at the start of main():

```
cout << std::fixed;
cout << std::setprecision(2);
```

When computing the # of students taught, simply sum the # of students in each course; the Course class contains a method that returns the # of students. The # of courses taught includes all types of courses (Letter, Satisfactory, and Unknown).

Once the summary is output, the program prompts the user to input a command. The user can input commands in any order, and can input any number of commands. The "#" command ends the program. There are 5 commands: "summary", "search", "unknown", "dfw", and "letterA". Commands are case-sensitive, and any other input should result in the output "**unknown command". Here's an example:

```
Enter a command> help
**unknown command
Enter a command> SUMMARY
**unknown command
Enter a command> search
dept name, or all? CS
course # or instructor prefix? 499
CS 499 (section 1): Hummel
 # students: 84
 course type: satisfactory
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
Enter a command> #
```

1. summary

The summary command outputs a college-wide summary, or a departmental summary. The user is prompted for a department name, e.g. “CS”, or “all” which means the college. The input is case-sensitive. Here’s a departmental summary:

```
Enter a command> summary
dept name, or all? CS
CS:
 # courses taught: 56
 # students taught: 4298
 grade distribution (A-F): 39.99%, 32.94%, 14.95%, 6.07%, 6.05%
 DFW rate: 17.64%
Enter a command>
```

Here’s a college-wide summary, note the departments are listed in alphabetical order by name:

```
Enter a command> summary
dept name, or all? all
BIOE:
 # courses taught: 23
 # students taught: 786
 grade distribution (A-F): 55.16%, 29.37%, 10.32%, 3.58%, 1.58%
 DFW rate: 7.80%
CHE:
 # courses taught: 16
 # students taught: 550
 grade distribution (A-F): 51.12%, 33.13%, 10.84%, 3.27%, 1.64%
 DFW rate: 7.92%
```

.
. **// more departments in alphabetical order by department name**
.

```
IT:
 # courses taught: 1
 # students taught: 24
 grade distribution (A-F): 33.33%, 20.83%, 16.67%, 12.50%, 16.67%
 DFW rate: 45.16%
ME:
 # courses taught: 39
 # students taught: 2154
 grade distribution (A-F): 45.75%, 28.01%, 18.34%, 5.58%, 2.32%
 DFW rate: 9.60%
Enter a command>
```

2. search

The search command performs a department or college-wide search by course number or instructor prefix. The user is first prompted for a department name, e.g. "CS", or "all" which means the college. The user is then prompted to enter a course number or instructor prefix. All input is case-sensitive. Here's a departmental search for CS 341:

```
Enter a command> search
dept name, or all? CS
course # or instructor prefix? 341
CS 341 (section 1): Grad Asst
 # students: 16
 course type: letter
 grade distribution (A-F): 50.00%, 35.71%, 7.14%, 0.00%, 7.14%
 DFW rate: 7.14%
CS 341 (section 2): Hummel
 # students: 134
 course type: letter
 grade distribution (A-F): 33.33%, 40.15%, 16.67%, 7.58%, 2.27%
 DFW rate: 11.19%
Enter a command>
```

And a departmental search by instructor prefix "Bel":

```
Enter a command> search
dept name, or all? CS
course # or instructor prefix? Bel
CS 151 (section 1): Bello Lander
 # students: 269
 course type: letter
 grade distribution (A-F): 39.93%, 27.24%, 17.16%, 7.09%, 8.58%
 DFW rate: 19.86%
CS 342 (section 1): Bell
 # students: 164
 course type: letter
 grade distribution (A-F): 31.29%, 55.21%, 10.43%, 1.23%, 1.84%
 DFW rate: 9.20%
CS 418 (section 1): Bello Lander
 # students: 59
 course type: letter
 grade distribution (A-F): 88.14%, 8.47%, 3.39%, 0.00%, 0.00%
 DFW rate: 1.67%
CS 440 (section 1): Bell
 # students: 114
 course type: letter
 grade distribution (A-F): 48.67%, 34.51%, 10.62%, 5.31%, 0.88%
 DFW rate: 6.19%
Enter a command>
```

Notice in both cases the output is by course #, then section #. Since the user can enter a course number or an instructor prefix, your app needs to determine whether the input is numeric (and convert if so). There are various way to do this, here's one approach using the C++ **stringstream** class. First, you need to #include <string> and <sstream>. Then

```
    string instructorPrefix;
    int    courseNum;

    cout << "course # or instructor prefix? ";
    cin >> instructorPrefix;

    stringstream ss(instructorPrefix);  // create stringstream object
    ss >> courseNum;  // try to convert input to a course #:

    if ( ss.fail() )
      // conversion failed, input is not numeric
    else
      // conversion worked, courseNum contains numeric value
```

Here are college-wide searches by course number and instructor prefix. Note that for college-wide searches, the output is sorted first by department, then course #, then section # (for a course # search), and by instructor, then course #, then section # (for a prefix search).

```
Enter a command> search
dept name, or all? all
course # or instructor prefix? 599
CS 599 (section 1): Yu
 # students: 16
 course type: satisfactory
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ME 599 (section 1): Shahbazian-Yassar
 # students: 11
 course type: unknown
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
Enter a command>
```

```
Enter a command> search
dept name, or all? all
course # or instructor prefix? Re
CS 107 (section 1): Reckinger
 # students: 110
 course type: letter
 grade distribution (A-F): 29.09%, 35.45%, 20.00%, 7.27%, 8.18%
 DFW rate: 29.01%
CS 109 (section 1): Reckinger
 # students: 253
 course type: letter
 grade distribution (A-F): 40.40%, 25.20%, 17.60%, 8.80%, 8.00%
 DFW rate: 24.36%
CS 110 (section 1): Reckinger
 # students: 48
 course type: letter
 grade distribution (A-F): 52.08%, 29.17%, 6.25%, 6.25%, 6.25%
 DFW rate: 12.50%
IE 342 (section 1): Reckinger
 # students: 131
 course type: letter
 grade distribution (A-F): 25.95%, 41.22%, 19.85%, 10.69%, 2.29%
 DFW rate: 16.79%
```

Continued on next page…

```
CME 315 (section 1): Reddy
 # students: 42
 course type: letter
 grade distribution (A-F): 42.86%, 45.24%, 9.52%, 2.38%, 0.00%
 DFW rate: 2.38%
CME 516 (section 1): Reddy
 # students: 17
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 10.53%
CS 141 (section 1): Reed
 # students: 209
 course type: letter
 grade distribution (A-F): 28.71%, 27.75%, 23.92%, 11.96%, 7.66%
 DFW rate: 26.32%
ECE 396 (section 2): Revelo Alonso
 # students: 101
 course type: letter
 grade distribution (A-F): 86.00%, 14.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ECE 397 (section 2): Revelo Alonso
 # students: 10
 course type: letter
 grade distribution (A-F): 90.00%, 0.00%, 10.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ECE 458 (section 1): Revelo Alonso
 # students: 36
 course type: letter
 grade distribution (A-F): 25.00%, 30.56%, 30.56%, 13.89%, 0.00%
 DFW rate: 16.22%
Enter a command>
```

3. unknown

The unknown command performs a department or college-wide search for all courses whose Grading type is Course::Unknown. The user is prompted for a department name, e.g. “CS”, or “all” which means the college. Here are the unknown courses for the ME department (output ordered by course #, then section #):

```
Enter a command> unknown
dept name, or all? ME
ME 211 (section 1): Megaridis
 # students: 74
 course type: unknown
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ME 599 (section 1): Shahbazian-Yassar
 # students: 11
 course type: unknown
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
```

Here are the unknown courses college-wide (output is ordered by dept name, course #, section #):

```
Enter a command> unknown
dept name, or all? all
BIOE 102 (section 1): Shokuhfar
 # students: 60
 course type: unknown
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
CS 474 (section 2): Deitz
 # students: 36
 course type: unknown
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ME 211 (section 1): Megaridis
 # students: 74
 course type: unknown
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ME 599 (section 1): Shahbazian-Yassar
 # students: 11
 course type: unknown
 grade distribution (A-F): 0.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
Enter a command>
```

4. dfw

The dfw command performs a department or college-wide search for all courses whole DFW rate exceeds a threshold input by the user. Here's an example for the CS department with a threshold of 40.0%, notice the output is in descending order by DFW rate:

```
Enter a command> dfw
dept name, or all? CS
dfw threshold? 40.0
CS 251 (section 1): Lillis
 # students: 170
 course type: letter
 grade distribution (A-F): 17.65%, 12.94%, 24.12%, 11.18%, 34.12%
 DFW rate: 51.56%
CS 425 (section 1): Kenyon
 # students: 22
 course type: letter
 grade distribution (A-F): 15.79%, 26.32%, 26.32%, 21.05%, 10.53%
 DFW rate: 48.00%
CS 251 (section 2): Troy
 # students: 108
 course type: letter
 grade distribution (A-F): 18.52%, 28.70%, 17.59%, 6.48%, 28.70%
 DFW rate: 40.68%
Enter a command>
```

If 2 courses have the same DFW rate, the output is then ordered by dept, course #, section # (all in ascending order). Here's a college-wide search for courses with a DFW rate above 40.0%:

```
Enter a command> dfw
dept name, or all? all
dfw threshold? 40.0
CS 251 (section 1): Lillis
 # students: 170
 course type: letter
 grade distribution (A-F): 17.65%, 12.94%, 24.12%, 11.18%, 34.12%
 DFW rate: 51.56%
CME 201 (section 1): Ataei
 # students: 143
 course type: letter
 grade distribution (A-F): 17.02%, 18.44%, 19.86%, 21.28%, 23.40%
 DFW rate: 48.00%
CS 425 (section 1): Kenyon
 # students: 22
 course type: letter
 grade distribution (A-F): 15.79%, 26.32%, 26.32%, 21.05%, 10.53%
 DFW rate: 48.00%
IT 202 (section 1): Hayes
 # students: 24
 course type: letter
 grade distribution (A-F): 33.33%, 20.83%, 16.67%, 12.50%, 16.67%
 DFW rate: 45.16%
ECE 465 (section 1): Dutt
 # students: 17
 course type: letter
 grade distribution (A-F): 25.00%, 31.25%, 25.00%, 18.75%, 0.00%
 DFW rate: 40.91%
CS 251 (section 2): Troy
 # students: 108
 course type: letter
 grade distribution (A-F): 18.52%, 28.70%, 17.59%, 6.48%, 28.70%
 DFW rate: 40.68%
Enter a command>
```

5. letterA

The dfw command performs a department or college-wide search for all courses whole percentage of letter A grades exceeds a threshold input by the user. Here's an example for the CS department with a letter A threshold of 80.0%. Notice the output is in descending order the percentage of A grades, where if 2 courses have the same percentage, the output is then ordered by dept, course #, section # (all in ascending order)

```
Enter a command> letterA
dept name, or all? CS
letter A threshold? 80.0
CS 587 (section 1): Solworth
 # students: 10
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
CS 594 (section 1): Petrov
 # students: 29
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
CS 596 (section 1): Ohlsson
 # students: 17
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
CS 502 (section 1): Berger-Wolf
 # students: 13
 course type: letter
 grade distribution (A-F): 92.31%, 7.69%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
CS 594 (section 2): Ziebart
 # students: 10
 course type: letter
 grade distribution (A-F): 90.00%, 10.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
CS 514 (section 1): Gmytrasiewicz
 # students: 29
 course type: letter
 grade distribution (A-F): 89.66%, 10.34%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
CS 418 (section 1): Bello Lander
 # students: 59
 course type: letter
 grade distribution (A-F): 88.14%, 8.47%, 3.39%, 0.00%, 0.00%
 DFW rate: 1.67%
Enter a command>
```

Continued on the next page…

Here’s an example of a college-wide search for courses above 99.0%:

```
Enter a command> letterA
dept name, or all? all
letter A threshold? 99.0
BIOE 525 (section 1): Lee
 # students: 13
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 7.14%
BIOE 562 (section 1): Mathew
 # students: 22
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
BIOE 576 (section 1): Esmailbeigi
 # students: 9
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 10.00%
CHE 425 (section 1): Liu
 # students: 12
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 7.69%
```

.
. **// more departments in alphabetical order by department name**
.

```
ENGR 194 (section 2): Abiade
 # students: 12
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ME 496 (section 1): Komperda
 # students: 10
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
ME 562 (section 1): Mathew
 # students: 12
 course type: letter
 grade distribution (A-F): 100.00%, 0.00%, 0.00%, 0.00%, 0.00%
 DFW rate: 0.00%
Enter a command>
```

Getting started…

As always, build the program slowly, one feature at a time. Note that one of the requirements will be to build on the GradeUtil functions developed in project 02 --- these functions will do most of the work needed in this program. That said, feel free to extend the GradeUtil classes and functions as needed. For example, you may find it helpful to add methods to the Dept and College classes, such as **getNumClasses()**.

If you did not complete project 02, our solution is being provided on the course web page: look under "Projects", "project02-solution":

https://www.dropbox.com/sh/hj59ao3hhqclnlw/AACnz9lznpGTq5CMrpVcDBTza?dl=0

The input file shown in the screenshots, "fall-2018.csv", is also provided on the course web page: look under "Projects", "project03-files":

https://www.dropbox.com/sh/mq2cfvl9uajzyum/AAAl8x0q04VXFgB66FAeYKHIa?dl=0

Just in case you're curious, the provided data is the actual data from the College of Engineering for Fall, 2018.

In terms of programming environment, you are free to work in the environment of your choice. If you prefer, you can also continue to use Codio and the "project01_testing" workspace. If you plan to use Codio, I would recommend creating a sub-directory and then moving all your files into that sub-directory for safe-keeping:

```
mkdir backup
mv * ./backup
```

This way you start with an empty workspace. You can copy files from the backup folder up into your workspace to get started, such as gradeutil.h and gradeutil.cpp:

```
cd backup
cp gradeutil.* ..
cd ..
```

Now create a new "main.cpp" file for your app, and a makefile to compile and run. Remember that when creating a makefile, you must indent using TABs, not spaces --- in Codio's View menu, uncheck "Soft Tabs" when editing your makefile so Codio does not turn your TABs into spaces.

Requirements

As is the case with all assignments in CS 341, how you solve the problem is just as important as simply getting the correct answers. You are required to solve the problem the "proper" way, which in this case means using modern C++ techniques: classes, objects, built-in algorithms and data structures, lambda expressions, etc. It's too easy to fall back on existing habits to just "get the assignment done."

In this assignment, your solution is required to do the following:

- Use the **GradeUtil** functions developed in project 02; use your own functions or the ones we have provided.
- All sorting must be done using **std::sort** (or another built-in sort). No explicit sorting.
- Use **ifstream** to open / close the input file for you. Be sure to check that the file was opened successfully since C++ does not check for you; if (!file.good()) is one approach.
- Use **range-based for** loops ("foreach"), there's no reason to use C-style index-based loops in this assignment.
- Define at least **3 functions** of your own design, and call them. These are in addition to the functions in the GradeUtil library. Nothing fancy required, just practice modular development.
- Use **standard C++** only; no third-party libraries.
- **No global variables**; use parameter passing to/from your functions.
- **No explicit pointers**; use references and iterators as appropriate.

For online documentation, I typically use the site http://www.cplusplus.com/reference/. Another way is to google with the prefix **std::**, which refers to the C++ standard library. Example: for information about the C++ string class, google "std::string". For the vector class, google "std::vector".

Electronic Submission and Grading

Grading will be based on the correctness of your console application in response to user input. We are not concerned with efficiency at this point, only correctness. However, gross inefficiencies are subject to deductions (e.g. a 5-level nested loop to find courses is *not* acceptable). Note that we will test your app against other input files with the same format, but different data.

When you are ready to submit your program for grading, login to Gradescope and upload your program files (.h and .cpp); you can upload the files directly, or in the format of a .zip file exported from Codio (Project menu, Export as Zip). If you have not yet registered on Gradescope, you can self-register at www.gradescope.com using the entry code 9DPNZ3. You have unlimited submissions, and Gradescope keeps a complete history of all submissions you have made. By default, Gradescope records the score of your last submission, but if that score is lower, you can click on "Submission history", select an earlier score, and click "Activate" to select it. The activated submission will be the score that gets recorded, and the submission we grade. If you submit on-time and late, we'll grade the last submission (the late one) unless you activate an earlier submission.

The grade reported by Gradescope will be a tentative one. After the due date, submissions will be manually reviewed for style, commenting, readability, and meeting of requirements. Correctness will account for 90% of the grade, and the remaining 10% dedicated matters of style, commenting and readability. Failure to meet a requirement --- e.g. use of std::sort --- will trigger a deduction of some kind.

Policy

Late work *is* accepted. You may submit as late as 48 hours after the deadline for a penalty of 10%. After 24 hours, no submissions will be accepted.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them (e.g. your "iClicker teammates"), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml .