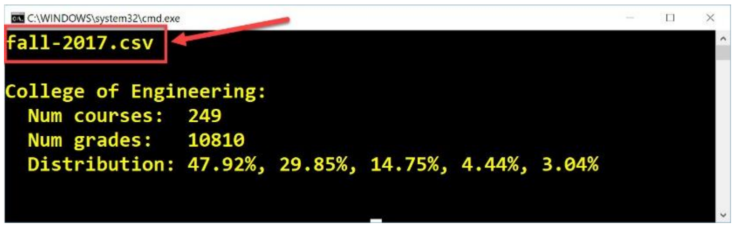
Project #01: Grade Distribution Analysis

Complete By: Tuesday, Sept. 11th @ 11:59pm (both sections)
Assignment: C++ program to perform Grade Analysis
Policy: Individual work only, late work *is* accepted (see "Policy" section on last page for more details)
Submission: online via repl.it (exercise **P01 Grade Analysis**)

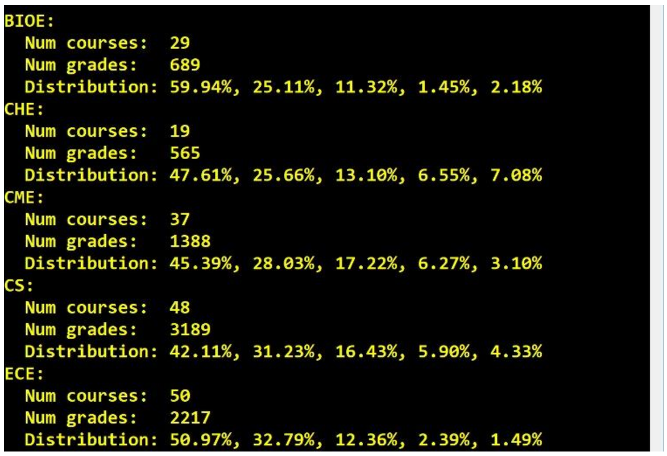
Assignment



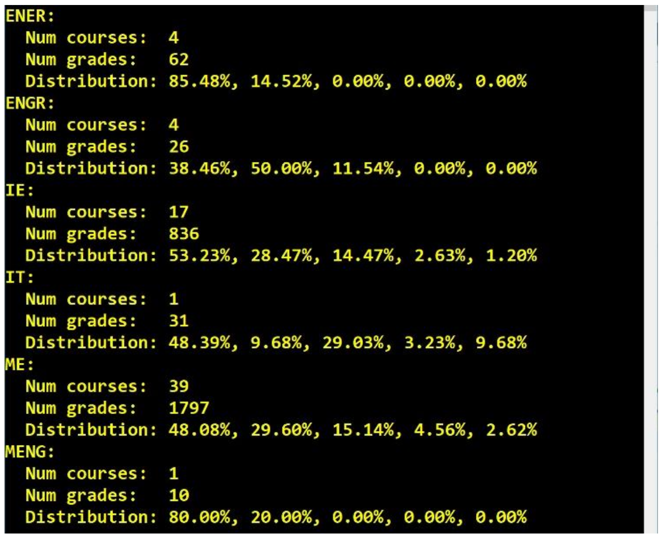
```
C:\WINDOWS\system32\cmd.exe
fall-2017.csv

College of Engineering:
  Num courses:  249
  Num grades:   10810
  Distribution: 47.92%, 29.85%, 14.75%, 4.44%, 3.04%
```

The assignment is to input grade distribution data for a given semester, and then output a variety of analyses. This is actual data for the College of Engineering, collected by UIC's Office of Institutional Research (http://oir.uic.edu/). The assignment comes in 4 parts. Given a filename input by the user, part 1 is to compute and output the grade distribution for the entire College of Engineering. Shown above is the output for Fall of 2017. The grade distribution is the percentage of A's, B's, C's, D's and F's assigned across all classes in the college; percentages are computed based on the total # of A, B, C, D and F grades --- all other grades (e.g. S and U) are ignored.

Part 2 is to compute and output grade distributions for each department offering courses. In the Fall of 2017 there are 11 such departments: BIOE, CHE, CME, CS, ECE, ENER, ENGR, IE, IT, ME, and MENG. Here are the corresponding grade distributions for Fall of 2017:

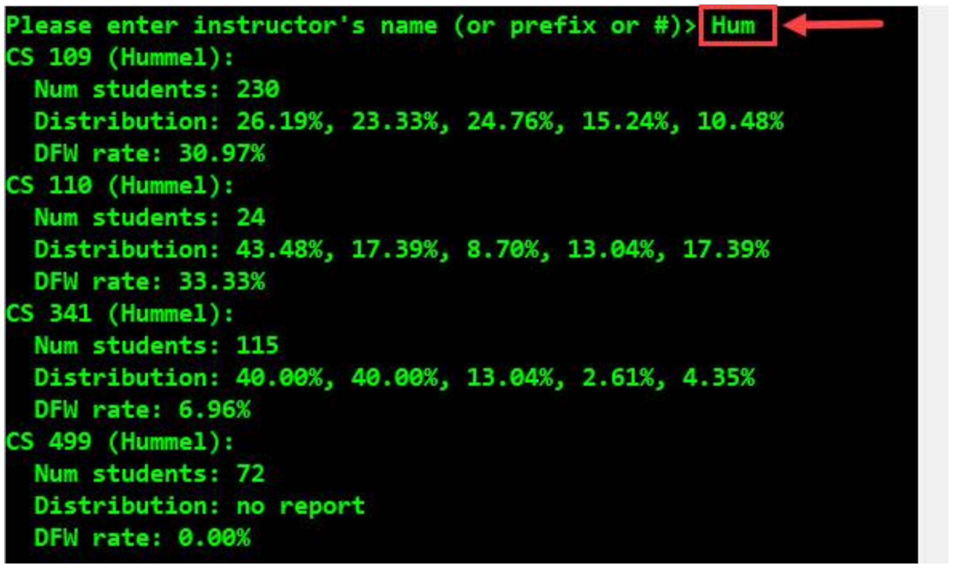


```
BIOE:
  Num courses:  29
  Num grades:   689
  Distribution: 59.94%, 25.11%, 11.32%, 1.45%, 2.18%
CHE:
  Num courses:  19
  Num grades:   565
  Distribution: 47.61%, 25.66%, 13.10%, 6.55%, 7.08%
CME:
  Num courses:  37
  Num grades:   1388
  Distribution: 45.39%, 28.03%, 17.22%, 6.27%, 3.10%
CS:
  Num courses:  48
  Num grades:   3189
  Distribution: 42.11%, 31.23%, 16.43%, 5.90%, 4.33%
ECE:
  Num courses:  50
  Num grades:   2217
  Distribution: 50.97%, 32.79%, 12.36%, 2.39%, 1.49%
```



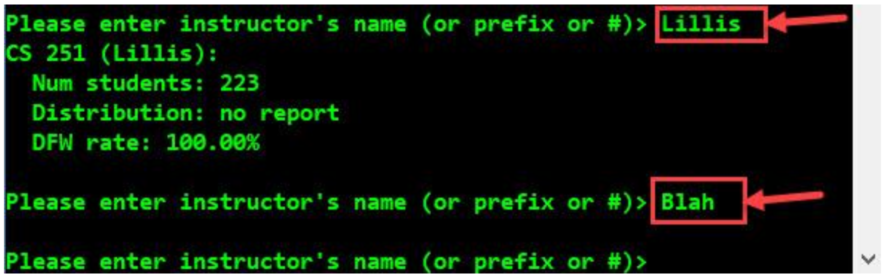
```
ENER:
  Num courses:  4
  Num grades:   62
  Distribution: 85.48%, 14.52%, 0.00%, 0.00%, 0.00%
ENGR:
  Num courses:  4
  Num grades:   26
  Distribution: 38.46%, 50.00%, 11.54%, 0.00%, 0.00%
IE:
  Num courses:  17
  Num grades:   836
  Distribution: 53.23%, 28.47%, 14.47%, 2.63%, 1.20%
IT:
  Num courses:  1
  Num grades:   31
  Distribution: 48.39%, 9.68%, 29.03%, 3.23%, 9.68%
ME:
  Num courses:  39
  Num grades:   1797
  Distribution: 48.08%, 29.60%, 15.14%, 4.56%, 2.62%
MENG:
  Num courses:  1
  Num grades:   10
  Distribution: 80.00%, 20.00%, 0.00%, 0.00%, 0.00%
```

Finally, part 3 is to interact with the user to allow the searching for grade distributions for courses taught by individual faculty. The user can enter a faculty member's last name (e.g. "Hummel"), or the start of his/her name (e.g. "Hum"). Here's the Fall 2017 output for Prof. Hummel:



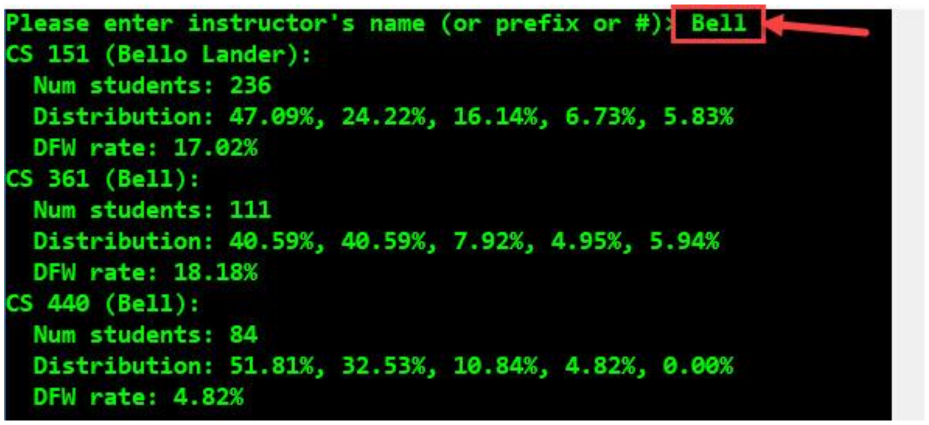
```
Please enter instructor's name (or prefix or #)> Hum
CS 109 (Hummel):
  Num students: 230
  Distribution: 26.19%, 23.33%, 24.76%, 15.24%, 10.48%
  DFW rate: 30.97%
CS 110 (Hummel):
  Num students: 24
  Distribution: 43.48%, 17.39%, 8.70%, 13.04%, 17.39%
  DFW rate: 33.33%
CS 341 (Hummel):
  Num students: 115
  Distribution: 40.00%, 40.00%, 13.04%, 2.61%, 4.35%
  DFW rate: 6.96%
CS 499 (Hummel):
  Num students: 72
  Distribution: no report
  DFW rate: 0.00%
```

The # of students is the total of all grades assigned. Note that for CS 499, Prof. Hummel has a grade distribution of "no report" because no letter grades were assigned (CS 499 is graded S/U). The grade distribution is computed as discussed earlier; the "DFW rate" is the percentage of students who received a D, an F, or withdrew (over the total # of A, B, C, D, F and W grades). Here's the Fall 2017 output for Prof. Lillis:



```
Please enter instructor's name (or prefix or #)> Lillis
CS 251 (Lillis):
  Num students: 223
  Distribution: no report
  DFW rate: 100.00%

Please enter instructor's name (or prefix or #)> Blah

Please enter instructor's name (or prefix or #)>
```

Note that if the user inputs a faculty name or prefix that doesn't match, your program should produce no output. Finally, here's the output for "Bell", which matches both Professors Bell and Bello Lander:



```
Please enter instructor's name (or prefix or #)> Bell
CS 151 (Bello Lander):
  Num students: 236
  Distribution: 47.09%, 24.22%, 16.14%, 6.73%, 5.83%
  DFW rate: 17.02%
CS 361 (Bell):
  Num students: 111
  Distribution: 40.59%, 40.59%, 7.92%, 4.95%, 5.94%
  DFW rate: 18.18%
CS 440 (Bell):
  Num students: 84
  Distribution: 51.81%, 32.53%, 10.84%, 4.82%, 0.00%
  DFW rate: 4.82%
```

The program ends when the user inputs **#**. Since most of you are relatively new to modern C++, approach this like a CS 1 assignment: one step at a time. Also, since the program will be auto-graded using the **repl.it** system, your output must match what is shown exactly. For example, notice the output is in order by Dept and Course Number.

Programming Environment

To facilitate testing and program submission, we'll be using the repl.it system; see the exercise titled "**P01 Grade Analysis**". You can work entirely within repl.it, or in the C++ environment of your choice; Visual Studio is not required for this assignment. Any environment with a modern (C++11 or newer) compiler will suffice; note that **bert** and **ernie** do *not* support modern C++. If using g++, you may need to supply the compiler option "-std=c++11" or "-std=c++14" to enable modern C++ support.

Input Files

The program starts by inputting the filename for a semester of course data, such as "fall-2017.csv". The input file is a CSV file that contains a header row followed by 1 or more rows of data. Here's the start of the "fall-2017.csv" file:

```
Dept,Number,Title,A,B,C,D,F,I,NR,S,U,W,Instructor
BIOE,101,Intro to Bioengineering,22,8,2,1,0,1,0,0,0,5,Eddington
BIOE,101,Intro to Bioengineering,12,13,8,0,3,0,0,0,0,3,Felder
BIOE,102,Bioengineering Freshman Smnr,0,0,0,0,0,0,56,0,0,1,Shokuhfar
BIOE,205,Biothermodynamics,20,24,7,0,0,0,0,0,0,4,Ma
.
.
.
MENG,400,Engineering Law,8,2,0,0,0,0,0,0,0,0,Rockman
```

Each line consists of 14 values, as denoted by the header row. You may assume the file is in order by Dept and Course Number. "I" refers to the number of incompletes, and "NR" the number of "no reports" --- e.g. students who never attended class and thus no grade could be reported.

The "fall-2017.csv" input file is available on the repl.it system, as well as the course web page under "Projects", then "project01-files". Note that the best way to download from the course web page is to use the "download" button in the upper-right corner of the underlying dropbox page.

Getting Started

Recall that we presented a complete C++ program on the first day of class; this program demonstrated a simple class, file input, parsing a CSV file, and use of the vector<T> class. The lecture notes are available on the course web page (PDF, PPT). To help you get started, a minimal "main.cpp" file is provided with the appropriate #includes; see the dropbox folder "project01-files".

As you design your initial solution, be sure to solve for the general case --- do not write a program that works only for the input file "fall-2017.csv". We will test your program against other input files of a similar format.

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 **ifstream** to open / close the input file; see the lecture notes from the first day of class (PDF, PPT) for examples of using ifstream, getline(file, line) to read input line by line, and **stringstream** to parse each line. Don't forget to skip the first line of input data, since the first line is a header row.
- Use one or more classes to store the input — at a minimum you must have a **Course** class that contains data about each course: dept, number, grades, etc. The design of that class is up to you, but at the very least the class must contain (1) a constructor for initializing its data members, and (2) at least 3 methods called by your program (e.g. you Course class might have methods for "percentageDFW( )" and "getNumStudents( )"). For simplicity, place your class definition(s) at the top of your "main.cpp" source file.
- Use **std::vector<T>** to store your objects; you are required to input the data exactly once and store into a vector of objects. While other containers might be more efficient, you'll have a chance to use those in later assignments. In this project, we want everyone to use std::vector<T>. Don't worry too much about efficiency, focus on learning modern C++.
- Whenever possible, use **range-based for** (aka "foreach") instead of index-based for loops.
- Use **std::sort** if you need to sort the data. Feel free to use other built-in functions as well, e.g. **std::find_if** for searching.
- No explicit pointers of any kind — no char *, no NULL, no C-style pointers
- No global variables whatsoever; use functions and parameter passing.

Writing code that is not modern C++ will result in a score of 0. Using a container other than std::vector? 0. No classes? 0. Using global variables? 0.

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".

Have a question? Use Piazza, not email

Do not email the staff with questions — use Piazza. Remember the guidelines for using Piazza:

1. *Look before you post* — the main advantage of Piazza is that common questions are already answered, so search for an existing answer before you post a question. Posts are categorized to help you search, e.g. "Pre-class" or "HW".

2. Post publicly — only post privately when asked by the staff, or when it's absolutely necessary (e.g. the question is of a personal nature). Private posts defeat the purpose of piazza, which is answering questions to the benefit of everyone.

3. Ask pointed questions — do not post a big chunk of code and then ask "help, please fix this". Staff and other students are willing to help, but we aren't going to type in that chunk of code to find the error. You need to narrow down the problem, and ask a pointed question, e.g. "on the 3rd line I get this error, I don't understand what that means…".

4. Post a screenshot — sometimes a picture captures the essence of your question better than text. Piazza allows the posting of images, so don't hesitate to take a screenshot and post; see http://www.take-a-screenshot.org/ .

5. Don't post your entire answer — if you do, you just gave away the answer to the ENTIRE CLASS. Sometimes you will need to post code when asking a question --- in that case post only the fragment that denotes your question, and omit whatever details you can. If you must post the entire code, then do so privately --- there's an option to create a private post ("visible to staff only").

Submission

The program is to be submitted via http://repl.it: see the exercise "**P01 Grade Analysis**". The grading scheme for this assignment is based 100% on correctness --- you must pass all test cases to obtain a score of 100. We do, however, expect basic commenting, indentation, and readability. You should also be using functions, parameter passing, and good naming conventions. Add your name to the header comment at the top of each .cpp and .h file.

When submitting on repl.it, be sure to run the test cases and THEN click the submit button. If you fail one or more test cases, be sure to "SUBMIT ANYWAY" so we have a submission to grade.

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 10%, and as late as 48 hours after the deadline for a penalty of 20%. After 48 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 .