Compilers

Some history and advice as we begin our adventure.
Some Final Grades

Note: There were no "+" grades. Weird. Why?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A-</th>
<th>A-</th>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>B-</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>C-</td>
<td>C-</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>D</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>2</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table shows the distribution of grades, with 'A' being the highest and 'W' being the lowest. The note indicates there were no '+' grades.
Some Final Grades

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A-</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2</td>
<td>B-</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>C-</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>D</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>2</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There will be no more incompletes.

Don’t even ask.
# More Final Grades

<table>
<thead>
<tr>
<th>Grade</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>5</td>
</tr>
</tbody>
</table>

Legend:
- A: A
- B: B-
- C: C-
- D: D
- F: F
- W: W
- I: I
- C+: C+
# Still More Final Grades

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>A-</td>
<td>A-</td>
<td>A-</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>B-</td>
<td>B-</td>
<td>B-</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>C-</td>
<td>C-</td>
<td>C-</td>
<td>C</td>
<td>C+</td>
<td>C+</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>11</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

CMPT 432 - Compilers
Do Not Make Excuses

- This is not acceptable:

   ----- Original message -----  
   To: Alan Labouseur/FAC/Marist@Marist  
   Subject: RE: Projects  

   Hi Professor Labouseur,

   I just looked at my email this morning, I'm surprised you didn't get my work. I thought I pushed it Sunday, But I now realize I didn't set upstream origin for the branch, so it didn't actually get pushed. I've just re-pushed this morning so you can review it.

- I did not “review it”. The grade remained the same.

- If you cannot figure out GitHub, you do not deserve a Computer Science degree.

- Speaking of GitHub . . . .
GitHub Commit Graphs

A and A-
GitHub Commit Graphs

A and A−
GitHub Commit Graphs

B and B-
GitHub Commit Graphs

C and C-

CMPT 432 - Compilers
GitHub Commit Graphs
GitHub Commit Graphs

F

CMPT 432 - Compilers
Advice

- You **must** have already taken Algorithms. Seriously.
- Attend every class. Don’t miss even one.
- Take notes in class, in pencil, on paper.
- Rewrite your class notes into a new notebook as a form of studying.
- Do every lab. Write them up in LaTeX.
- Work on your projects every day. Commit to GitHub at least once a day, if not more frequently.
ADVICE from Dr. Helen Rothberg

- **Action** — Do more, say less. Pursue your interests. Ask for help when you need it.
- **Determination** — Stick to it and get things done.
- **Vision** — What are you trying to become? What do you want to do with your life? How will you get there?
- **Integrity** — Tell the truth all the time. Don't create or participate in drama.
- **Communication** — Be attentive and dig in, even if it's hard.
- **Empathy** — Dare to care about yourself and others.
Academic Honesty

- These are individual projects. All work must be your own.
- Any violation of this will result in your **immediate** and **automatic** failure of this class.
- This is your only warning. The next time we talk about this it will be in the context of me telling you that you have failed this class.
- I am serious. Ask prior students.
Academic Honesty

- I will be checking all your code against...
  - your class
  - other classes
  - all past projects
  - the hall of fame
  - and...

Past Project Hall of Fame
JavaScript or TypeScript projects only. There have been some fantastic projects in Scala, C++, Python, and other languages, but they (obviously) will not execute in a web browser.

2019
- Juice Compiler

2018
- Sonar
- Kompiler
- The Colorful Compiler
- Daniel Ahl's Compiler

Note: Earlier compilers used slightly different grammars compared to our current grammar.

2017
- Terminator

2016
- Although some were close, none of the JavaScript or TypeScript projects were worthy of this hall of fame. There were some excellent projects implemented in other languages.

2015
- Andrew B
- Bloop Compiler

2014
- Sugglator
- Rob W
- ChronOS/LL in OmniChron
Academic Honesty

- I will be checking all your code against...
  - your class
  - other classes
  - all past projects
  - the hall of fame
  - and...
  - GitHub
Academic Honesty

- I will be using MOSS to help detect cheaters. See theory.stanford.edu/~aiken/moss/

**Moss**

A System for Detecting Software Similarity

**UPDATES**

- Feb 1, 2018 And even more community contributions have been added!
- Nov 9, 2017 More community contributions have been added below ...
- Aug 31, 2017 Thanks to Christophe Troester for an OCaml client for Moss.
- May 18, 2014 Community contributions (including a Windows submission GUI from Shane May, thanks!) are now in their own section on this page.
- May 14, 2014 And here is a Java version of the submission script. Thanks to Bjørn Ziekel!
- May 2, 2014 Here is a PHP version of the submission script. Many thanks to Phillip Rehs!
- June 9, 2011 There were two outages over the last couple of days that lasted no more than a hour each (I think). I've made some changes to the disk management software that should prevent these problems from recurring.
- April 29, 2011 There was an outage lasting a few hours today, the first since last summer, but everything is back up.
- August 1, 2010 Everything is back to normal.
- July 27, 2010 The Moss server is back on line. There may be some more tuning and possibly downtime in the coming weeks, but any outages should be brief. New registrations are not yet working, but people with existing accounts can submit jobs.
- July 25, 2010 As many (many!) people have noticed, the Moss server has been down for all of July. Unfortunately the hardware failed while I was away on a trip. I am hopeful it will be back up within a few days.

What is Moss?

Moss (for Measure Of Software Similarity) is an automatic system for determining the similarity of programs. To date, the main application of Moss has been in detecting plagiarism in programming classes. Since its development in 1994, Moss has been very effective in this role. The algorithm behind moss is a significant improvement over other cheating detection algorithms (at least, over those known to us).
Write Original Code

- Infuse your code with your personality.
  - comments
  - variable names
  - code conventions and style
  - structure
  - modularization

- List collaborators, references, and sites consulted in a comment block at the top of each file.
Before asking me for help

- Review your code yourself by reading it from the perspective of someone seeing it for the first time.
  - Consider your variable names — are they descriptive?
  - Did you write plentiful comments about why you made those choices rather than what the code does?
  - Examine your code conventions and style — is it consistent?
  - Fix anything that’s missing, unclear, or confusing.
Before asking me for help

- Set a break point in your browser or IDE and step through your code line by line for several test cases.
  - This is the first thing I’m going to ask you to do, so you might as well do it preemptively.
  - You will be amazed at how many mistakes, bugs, and off-by-one errors you’ll find that way.
When asking me for help

- Explain the problem clearly. Context matters.
- Describe several test cases that illustrate the problem.
- Tell me what you observed when stepping through your code line by line.
- Make sure the latest version is committed on GitHub and include a link to it in your e-mail.