CSCA20 - Project

Let's Have Some Fun

Learning Objectives

We've spent the term learning how to write code to solve problems... so now we're going to write code to solve problems. Your job is to find the problems to solve.

Evaluation

Your project will be marked as follows:
Ambition /5
Execution /5
Cool Factor /1
Total Mark: ((Ambition * Execution) + Cool Factor) /25
So, for example: if you have an ambition grade of 3/5 and an execution grade of 4/5, and a cool factor grade of 1/5, your total mark would be:
(3 * 4) + 1 = 13/25

Ambition

This is a metric of how difficult a problem you are trying to solve with your project. A very simple project will have a low ambition mark and a very complex project will get a high ambition mark. As a rough rubric: An ambition score of:

- 1 is awarded to a project that is equal to the complexity of a lab, showing off some skills at the advanced/extension level
- 3 is awarded to a project that is equal to the complexity of 2-3 labs, showing off and combining multiple skills at the advanced/extension level
- 5 is award to a project that is going above and beyond what we have covered in class, bringing in tools and techniques beyond the extension level

Execution

This is a metric of how well your project actually works. A project that does not run will have a very low execution mark, a project that works flawlessly and is well designed/documented will have a high execution mark. A a rough rubric An execution score of:

- 1 is awarded to a project that works in general functionality
- 3 is awarded to a project that works, is well designed and documented, and has been tested for obvious boundary cases
- 5 is awarded to a project that is polished, with very clear documentation, very good use of functions and control structures, and where all components have been thoroughly tested

Cool Factor

Sometimes a project checks all the boxes in terms of code quality and style, but the idea is chosen to be easy and safe. Sometimes projects aren't quite as polished, but really set out to tackle a real world problem. So we want to give a small bonus to people who are doing something cool with their project: Building something that might be helpful to the world, solving a problem that exists beyond this course, just producing something that shows us they're passionate about the idea. Basically, if you can make the teaching team go: "That's cool", you get a bonus mark.

Multiplicative Scoring

The final score is calculated not by adding ambition + execution, but rather by multiplying ambition * execution. Why do we do this?

- It is relatively easy to get a score of 5/5 on ambition if you don't care about execution (I can say I'm going to build the most complex thing in the world and just not actually do it)
- It is relatively easy to get a score of 5/5 on execution if you don't care about ambition (I can make a program that prints the number 7, and it will always work in every test case)

- Multiplying the values together means that you need to balance execution and ambition and can't sacrifice one for the other
- This project is meant to be extension of core concepts covered in the course. The goal here is to get students to challenge themselves and go beyond the material covered in the course. So we want to reward students who really push themselves by taking on a challenging project and putting in the time and effort to test and polish their work.

Team Work

You may work in teams of up to 3 people. But keep a few things in mind:

- All team members are responsible for the entire project. You will all be expected to have contributed, and the TA may ask any team member to demonstrate or explain any part of the code.
- While a team of 3 may not necessarily be able to build something 3 times more complex than a single student, we will expect a slightly higher level of ambition from larger teams
- You are allowed to work on your own... but team design and coding is more productive (and frankly, more enjoyable) than working alone

External Tools

You are allowed to use outside sources for this project. In fact, this is encouraged. But you MUST clearly document both in your code and in your demos any code/tools/resources/outside help you received from any source that is not directly part of this course.

Demoing and Submission

In addition to the live demo, you will also be required to submit the following to Quercus before the demo session:

- Your code (including any external files needed to run your code)
- A text document with the following information:
  - The names & student numbers of all team members

  - Instructions on how to run your code and any imported modules it requires
  - A link to a video of you demonstrating the file (we will not be marking you based on the quality of the video, we just want the entire TA team to be able to see your code in action so we can be sure we're marking fairly, and also because we're excited to see all the cool projects)

Only one team member needs to submit to Quercus

Project Scope

Your project really can be anything (well... within reason... let's try to keep it within the bounds of legality/propriety). Whatever you're interested in, build something related to that. The goal is to take everything you've learned this year and use it to create something that you want to see existing in the world. This course has a lot of amazing people with a lot of cool passions... it's your turn to show off!