Project notes

These are some rules and guidelines that apply to all of the projects in this course.

Project domains

Every project handout starts with an overview that sets up the problem. It gives you information about the problem domain that you might not be familiar with, and it sets the goals and motivations for the project. In general, the handout will go out before I've covered all the material you'll need to *complete* the project, in part because I will use the ongoing project as a running example in class—which means you'll need to read it as soon as it's assigned, since I'll be talking about it right away.

Checkpoints

After the overview of the project topic, the next portions of each project handout will be checkpoint assignments. The first, which I refer to as "prep work", generally won't work as much with the core conceptual content of the project, but will give you a head start on things like input formats and data structures that you'll need. Which means you can (i.e. should) start on it right away, even before we've finished covering the related content in class.

The second checkpoint is the "design work", which is to be written on paper (or the equivalent) and brought to class: diagrams, test cases, and other planning work that will help you understand how the program you're writing will need to fit together. I am looking for good-faith best-effort work here, not perfection; we'll talk about everybody's design in class and improve them, so that you'll be well-prepared to really forge ahead on the main body of the work.

Timeline

As a general rule, projects will be initially assigned on a Thursday and due three weeks later. (Project 1 is slightly shorter.) The prep work will be due after the first week; they're due **by 8pm** on the due date. (This gives you a chance to ask questions in class and go back and fix things.)

The design work will generally be due on the following Tuesday, sometimes alongside other homework.

The final handin for a project is due also **by 8pm** on the due date.

Grading

The prep assignment is worth 15 points. You get 15 points if it compiles, runs, and it does at least approximately what it's supposed to. You get 10 points if it compiles, runs, and does something relevant to the task. You get 0 points if it doesn't compile, if it immediately crashes, or if I can't easily figure out how to make it work. (So, you should include some documentation.) If your prep work gets a zero, work with me ASAP and you can get back up to 10 of the 15 points. (The purpose is to get you started, not to create a penalty!)

The design work is worth ten points, and is graded for completion. If you're absent on the design day and haven't made some sort of advance arrangements, that will normally be a zero on the design work.

The final handin should also include documentation, and this is worth 25 points if there's enough documentation that I can *easily* see what to do with your code and what your code does. The more I have to work to figure out what your code even does, the lower this gets. If you have no documentation that I can find, you get zero points for documentation.

The remainder (100) of the points are reserved for project-specific points, of which you will get zero if your program doesn't compile or if it immediately crashes. If it does run, the points are calibrated so that (together with a full 50 from the checkpoints and documentation), you'll get

- a low D for not making it past the checkpoint
- a middle C for a program that does some relevant thing

- a high B for a program that avoids major pitfalls and correctly solves at least a simple version of the project problem
- a high A or full credit for a program that really gets into the conceptually interesting parts of the problem.

I'll put rubric specifics in each project handout. Note that I'm not planning to, in general, read your code or comment on it after you hand in—if you want feedback on your code, see me while you're working on it. I will assign scores based on reading your docs and running your code, but you should be able to predict your score within a few points based on the rubric I give you in advance. In fact, one way to ensure that you maximise your points is to use that rubric as a template for writing your documentation and *making sure you tell me/show me how your program meets each of the rubric requirements.*

Each project will be worth 15% of the final grade.

Collaboration and citation

The projects in this course are collaborative, meaning you can (and should!) discuss your ideas with other students, but the code you write needs to be your own. See my collaboration policy for many examples of what is and isn't acceptable.

The nature of this course and these projects also makes it more reasonable to look on the internet for explanations, algorithms, and even code examples. That's great! But it also makes it more important than ever that you cite your sources. Make sure you read my citation policy for when and how to do this for programs.

Handing in

There is a `handin` command on the lab machines that you will use to hand in all your work. If you use your own computer for development, transfer your files to the lab machines, *verify that they still work*, and then hand them in by typing

```
handin cmsc262 proj1 file1 file2 file3
```

or just

```
handin cmsc262 proj1 dirname/
```

to hand in an entire directory of files, replacing `proj1` with the actual name of the assignment.

Don't forget to include your documentation, and make sure it's easy for me to find it. `README.txt` is a great name for a file of documentation that you want me to be able to find.

Use the same project name for both the prep work and the final handin.