Project notes

These notes 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 portion of each project handout will detail two important early checkpoints: one "prep" checkpoint that will typically be some lightweight coding work to begin setting up the program, e.g. with boilerplate code or basic I/O, and a second "design" checkpoint that will be on-paper work to get you started planning the program layout.

Timeline

As a general rule, projects will be due three weeks after they are assigned. (Project 1 is slightly shorter.) The prep checkpoints will be due after the first week; **by 4pm** on the due date. (This gives you a chance to ask questions in class to resolve some last-minute details, but you won't want to wait to start them until then!) The design checkpoints are due the Tuesday after that; **in class** on the due date, either on paper or on a device you bring with you, because we'll be talking about and comparing designs.

After the second checkpoint there's no formal handed-in checkpoint assignment, but I'll check in with you during class to see how far you've gotten and if you have questions.

The final handin for a project is due also **by 4pm** 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.)

The design checkpoint is worth 10 points. It doesn't have to be correct but should represent a good-faith effort; while you are comparing notes with your classmates I will circulate around the class to check that you've done it. Absent students will normally not receive credit for the design checkpoint.

The final handin should also include documentation, and this is worth 25 points if there's enough documentation that I can quickly figure out what to do with your code and what you think your code does and whether it actually does that. 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 prep checkpoint
- a middle C for a program that does something generally in the area the project is meant to cover (AI, databases, etc)
- 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 conceptual area (AI, databases, etc) 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.

I'm grading all four projects out of 150 points, and each will count as 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 checkpoint and the final handin.