CMPSC 200: Course Contract

This document represents the `CMPSC 200` evaluative terms and the ethos that und erlies them. To anyone who just wants to see the mechanics of the contract, skip to the **Evaluation** section.

For anyone who wants a bit of history, read on.

Instructor's note

This course aims to give you an awareness of the *very* low-level control structures that underlie computer technology or, as expert jockeys of the trade would tell you: as close to *bare metal* as possible.

Much of the spirit that motivated the development of the modern computer came from folks both needing and wanting to be *very clever*. There are many good histories on the subject, but early computational history is as much one of desperation and necessity as it is of good planning, foreward thinking, and high-level design. After all, the first major challenge in digital computation was to *build a computer* which reduced to, as George Dyson tells us in *Turing's Cathedral: the Origins of the Digital Universe* about John von Neumann (i.e. "Johnny") and his co-conspirators:

> after .... consultation with [major parts manufacturers], whose response was .... "there were no reliable [vacuum] tubes, and no reliable resistors." The response .... was that "we would have to learn how to design a reliable [machine] with thousands of unreliable components." And they did.

To but it bluntly, we are where we are because of *hacks*. Many of these hacks rely on low-level knowledge to enable high-level functionality. I think it's important that we read a bit about the historical definition of the "hack" to understand where my interest in evaluting this course derives.

A brief history of "hacks"

> The Hacker Ethic:
>
> **Access to computers—and anything that might teach you something about the way the world works—should be unlimited and total. Always yield to the Hands-On Imperative!**
>
> Hackers believe that essential lessons can be learned about the systems—about the world—from taking things apart, seeing how they work, and using this knowledge to create new and even more interesting things. They resent any person, physical barrier, or law that tries to keep them from doing this. This is especially true when a hacker wants to fix something that .... is broken or needs improvement. Imperfect systems infuriate hackers, whose primal

> instinct is to debug them. This is one reason why hackers generally hate driving cars— the system of randomly programmed red lights and oddly laid out one-way streets cause delays .... so unnecessary that the impulse is to rearrange signs, open up traffic-light control boxes .... to redesign the entire system.
>
> In a perfect hacker world, anyone pissed off enough to open up a control box near a traffic light and take it apart to make it work better should be perfectly welcome to make the attempt. Rules that prevent you from taking matters like that into your own hands are too ridiculous to even consider abiding by. This attitude helped the [Technical] Model Railroad Club (TMRC) start, on an extremely informal basis, something called the Midnight Requisitioning Committee. When TMRC needed a set of diodes or some extra relays to build some new feature into The System, a few people would wait until dark and find their way into the places where those things were kept. None of the hackers, who were as a rule scrupulously honest in other matters, seemed to equate this with "stealing." A willful blindness.
>
> All information should be free.
>
> If you don't have access to the information you need to improve things, how can you fix them? A free exchange of information, particularly when the information was in the form of a computer program, allow ed for greater overall creativity. When you were working on a machine .... which came with almost no software, everyone would furiously write systems programs to make programming easier—Tools to Make Tools, kept in the drawer by the conso le for easy access by anyone using the machine. This prevented the dreaded, time-wasting ritual of reinventing the wheel: instead of everybody writing [their] own version of the same program, the best version would be available to everyone, and everyone would be free to delve into the code and improve on that. A world studded with feature-full programs, bummed to the minimum, debugged to perfection.
>
> Steven Levy, *Hackers: Heroes of the Computer Revolution*
>
> **Instructor's note**: The above book is authentic and historically accurate, but a record of problematic behaviors.

We come to our work in an era where "hacking" refers to malicious or nefarious work, but the essential meaning of the term doesn't entertain this idea as essentially evil; to early computer systems folks – many of whose names you know very well – the declaration to complete and total systems control and information was equal to an inalienable right. You've likely drawn a parallel between the above set of statements about systems to those that the discipline of computer science (at least in this department's view of it) holds sacred.

While we're not abridging trade secrets in this course – all the platforms and tools we're using are, essentially open source – the spirit of discovery, exploration, and adventure that characterizes early computing is what I hope to be the predominate impression you have of our work together this semester.

Evaluation

The `Basic Grade`

Students who meet the basic covenants of this contract will earn an individual letter grade of `C` in this course.

You may ask why the `Basic Grade` is a `C` and *not a `B` or `B-`*.

Rising to the level of above-average or exceptional work relies on the just that: above-average or exceptional work. This contract should provide incentives to think beyond mere sufficiency. As you'll see below, work that aspires to higher levels of reward involves exploring the limits and opportunities of the concepts this course explores.

"Hacks"

This course will propose a number of assignments whose main limitations are purely technical, meaning that some systems-level barrier will stand in the way of completing objectives. Overcoming these challenges relies on the basic knowledge represented by the course's learning objectives. *However*, there are some more-or-less clever ways ("Hacks") which can achieve our ends and demonstrate above-average proficiency in applying a given topic.

Each lab assignment will feature suggestions of Hacks to implement. You may choose one of these suggested Hacks or propose one of your own provided that:

- code comments document the area of code featuring the Hack
- accompanying documentation clearly and *gregariously* provides:
  - a description of the hack
  - a record of the
  - an explanation of the potential value of implementing the hack

"Clearly" An explanation of a Hack is "clear" if it:

- Unambiguously describes the functionality implemented at the level of individual or related groups of instructions, accounting for all instructions which represent the Hack
- Comprehensively discusses the code or procedure implemented with reference to learning materials accompanying the assignment
- Exhaustively documents any limitations or potential pitfalls of the Hack including, but not limited to:
  - misuses possible *because* of the Hack (e.g. buffer overflows)
  - test cases which consistently and reliably result in system failure

– assumptions which users must know before using the Hack

"Gregariously" Depending on assignment complexity, requirements for length of a response ("gregariousness") will vary. Each hack will feature a minimum word count associated in order to claim credit for the novel and/or clever method you've come up with. These counts will be provided as part of the assignment in a special section ("Hacks") dedicated to outlining the potential approaches and rules for implementing hacks.

Hack rewards The following table describes the letter grade a student can expect given labor spent on addressing assignment "hacks", given that that there are `10` major assignments in this course, not counting the course project.

Hacks awarded	Letter grade
6	A
5	A-
4	B+
3	B
2	B-
1	C+

Evaluation penalties

There are three (3) situations which adversely affect evaluation:

1. `Incomplete` assignments or course project (i.e. basic expectations are not fulfilled)
2. `Ignored` assignments or course project (i.e. little or no evidence of assignment attempt)
3. Plagiarism or other inappropriate/dishonest academic behavior

Each of these circumstances results in a different outcome.

`Incomplete` assignments or project `Incomplete`: more than 50% – but less than 100% – of GatorGrader criteria pass.

Assignments For every incomplete assignment, a student's course grade will be lowered by half a letter grade.

Project An `Incomplete` course project will result in a course grade of `C`.

`Ignored` assignments or project `Ignored`: no attempt made to `push` work to GitHub *or* more than 50% of GatorGrader criteria fail.

Assignments An `Ignored` assignment results in a student's ourse grade being lowered by one whole letter grade for each such assignment.

Project An `Ignored` course project will result in a course grade of `F`.

Plagiarism or other inappropriate/dishonest academic behavior Students engaging in academically dishonest behavior will be assessed on a case-by-case basis, as circumstances and factors of a case vary widely. However, in situations where academic dishonesty is established, the variety of penalties might include (but are not limited to):

- disqualification from receving credit for a Hack
- automatic assessment of an 'Ignored" assignment for a given assignment
- ineligibility to earn a grade above the `Basic Grade`

In severe cases, academic dishonesty may be referred to the Allegheny College Honor Code Committee, though these cases are rare (i.e. *almost* non-existent). For clarification regarding what qualifies as "academic dishonesty" consult the Syllabus and the *Compass*.

Note about assignment completion This agreement operates on the understanding that circumstances may arise which interfere with completing assignments on time. Should you need more time to complete work, arrange an assignment extension with the instructor.

Assignment "bugs" and impact on grading

- In the event that there are bugs with grader criteria, the instructor will assess assignment completion status with known issues in mind with the caveats that:
  - the instructor will make every effort to remedy the failing check before the assignment due date
  - if, during assignment evaluation, a check fails for every student due to a technical issue, the check is considered an invalid grading criteria

Policies

Course platforms

On course platforms used for the duration of the semester:

- Use all platforms for appropriate class reasons
- Follow the instructions while using the platforms
- Keeping "up-to-date" with platforms may require seeking additional extra-curricular knowledge
  - For example, using Markdown in Discord
- Respect others' thoughts, time, and contributions to either discussion or code products

- Check course platforms often – at least twice a day
  - The best times to review are often the beginning and ends of a day
    - i.e.: 8:00 (beginning), 18:00 (end)

Using office hours

When using office hours:

- Come in prepared with questions, a topic, or a clear objective
  - To achieve this, please add notes about expected topic(s) to a Google Calendar slot reservation (for instructor calendar)
- Be prepared to receive feedback, understanding that typically this means completing revisions on work discussed
- Have all assignments ready to discuss with the instructor
- Not rely only on office hours (i.e. work should be attempted first)

Technical Leaders

This course features the assistance of several peer educators that the department refers to as Technical Leaders (TLs). You should recognize these citizens as quasi-authorities in this course. While they don't make any kinds of grading decisions, they are all much smarter than the course instructor. They're also much cleverer.

These people are not course resources in the sense that they are inanimate machines. You agree to:

- Treat them courteously and with respect
- Not make strict demands of them; their engagement with the course is a privilege, not a right
- Recognize that their feedback won't give you the "answer"; they're excellent resources to help you *reason*
- Interaction and communication with TLs is functionally equivalent to interaction with the instructor

In the event that a conflict arises with a TL, please message the course instructor. Know that any issues that TLs encounter *are also reported to me*, and if we need to discuss your engagement with a TL, the instructor agrees to arbitrate the issue with fairness to all involved.