COM S 327, Spring 2023
Programming Project 1.10/2
Choose Your Own Assignment

This assignment is officially due on the date given on Canvas and in the syllabus. It is unofficially extended with no penalty through Friday of dead week. Voluntary demos for a small amount of extra credit will take place in the second-to-last lecture of the semester.

As discussed in class, the final assignment is something of your choosing. It should be of similar complexity to the weekly assignments throughout the semester. It can be an extension to the game, it may be something entirely standalone, or it may extend some other program. It should be in C++[1]

Your assignment should be of roughly the same complexity as the assignments we've had all semester. **Your assignment should not be boring!** Do something fun! Fun for you to write, and fun for the TAs to evaluate!

One option will be to implement effectively 1.10 of our pokémon game, fleshing out a few more mechanics and making it feel more complete. I used to give suggestions here on ways that you could do that, but last semester everybody[2] did exactly what I suggested, and then about a dozen of them presented it for EC in lecture; everybody else fell asleep. It was *not* fun. It was lazy and boring.

Unrelated to the game, I enjoy implementing recreational mathematics ideas; maybe you would too? Something like the Collatz Conjecture is certainly too simple, but you could write a program to render a number of fractals write them to image files (if you want to write images, I can supply you with some very simple, easy to use code for this), or allow infinite zoom through a fractal. You could create Mandelbrot sets, Julia sets, Sierpinski gaskets, etc.

Finite automata are fun, too. Again, most are too small on their own, but if you, e.g., encode output to video, that would probably be sufficient.

It's always fun to calculate pi in unusual ways. You may need a library for arbitrary precision (like the GNU MP Bignum library (GMP) or LiDIA (much more than just bignums)) to implement some of these.

Implement an encryption algorithm (also probably needs GMP).

Implement a extension to Angband or Nethack (a pair of very successful roguelike games). In either case, the work would be in C, not C++, and that would be okay, and I wouldn't expect a lot, because you'd spend the better part of the week just getting to know the code.

Some other things that students have done in the past:

- Implement a curses-based tron game.
- Implement a curses-based side-scroller (think: *Super Marie Bros.*).
- Implement a curses-based 3-D engine for your roguelike (think: *Escape from Castle Wolfenstein*[3]).
- Port your roguelike to Android or IOS.
- Implement a web interface for your roguelike.
- Crack RSA[4].

[1] I will entertain the prospect of other languages, but you'll need to discuss it with me in advance, and if the language is managed (i.e., does not require explicit memory management), I will reject it.

[2] Not literally everybody

[3] I'm serious. Somebody did this. And it was amazing.

[4] To be clear, the student didn't solve the general problem of cracking RSA. Instead, I implemented RSA and presented it in class. In class, I explained that my implementation uses a bad random number generator to calculate keys, and the student knew the time that I generated my keys within about ± 1 minute. The student also had access to an under-utilized compute cluster of a few hundred nodes and a couple of weeks to work. Still, *very* impressive.

- Implement a simple chess engine.
- Implement a simple droughts engine.
- Implement a simple reversi engine.
- Implement a simple connect four engine.

Implementing snake games was very popular in past semesters. So popular that the TAs and I have grown bored with them. Snake games will not be acceptable unless you make it somehow unique and interesting. Nobody wants yet another snake game.

This document is intended to provide *ideas*. This is *Choose Your Own Assignment*! You're welcome to do whatever you like here, so long as it fits within the rules given in the first couple of paragraphs.