COM S 3270, Spring 2025
Programming Project 1.10/2
Choose Your Own Assignment

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]

Two extensions to the game that are potentially–depending on design decisions–big enough to be an assignment:

1. Ranged combat. Add a command to select a target (cell or monster). A ranged weapon must be wielded (bow, sling, holy hand grenade, etc.), and a second command will attack that target as long as it remains valid. Also add a command to cast a poison ball spell. The spell centers on the target and damages all monsters in a radius around it. Add whatever other fun and clever extensions to this idea you like.

2. Update save and restore. We now have objects on the floor and in PC inventory and equipment. We also have monsters. The PC and monsters have a next turn, hitpoints, etc., and the dungeon has a character sequence number. Save all of this information to disk and reload the game from it so that it continues correctly.

Unrelated to the game, I enjoy implementing recreational mathematics ideas. 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. 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.

Below are a bunch of ideas related to the game. Most of them are too small by themselves, but could be extended or combined.

- Characters regenerate hitpoints at a rate of some percentage of their maximum hitpoints per game turn. Hint: You should not update this every game turn! That would be terribly inefficient. Instead, mark each character with a turn number that is the last time the character's HP were updated and update on demand.

- Add a command to allow the user to select cells in the dungeon and get a description of monsters and items there.

- Make lights work.

- Add spells, requiring spell books and "mana".

- Add spells and ranged attacks for monsters.

[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 have explicit memory management), I will reject it.

- Add other item effects, like resistance to elements, telepathy, ability to see invisible monsters (and add invisible monsters!)...
- Make dodge, defense, weight, and hit item attributes meaningful.
- Add meaningful dungeon levels which load more powerful monsters as you go down, easier monsters as you go up, perhaps a town with shops at the top.
- Add meaningful character statistics (strength, dexterity, constitution, intelligence, etc., maybe skills) and character levels.
- Make containers do something.
- Add new types of terrain, like water, lava, quicksand, etc., and make them affect gameplay in some sensible way.
- Add something that sounds fun and interesting to you.
- Develop your game into something complete enough to be interesting to the roguelike community (we're actually not that all that far away at this point) and release it. If you release something that gains users and continues development, it could be a very nice item on a resumé.

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*[2]).
- Port your roguelike to Android or IOS.
- Implement a web interface for your roguelike.
- Crack RSA[3].
- 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.

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

[3]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.