COM S 327, Spring 2017
Programming Project 1.06
"Fog of War" and porting to C++

We'll be converting (porting) our entire game into C++ this week. You'll need to rename all of your C files with a `.cpp` extension (if using my makefile, do a `make clobber` before renaming). You'll also need to update your makefile. If you're using one of my code drops (or if you've taken my makefile into your project), there is already automatic support to compile C++ code; however, you will need to change the link line to link with the C++ compiler (try to figure this out on your own).

All structs in your game should be changed to classes. There are no requirements about access to members (public, private, etc.); you decide what you think is best. My code uses an object oriented design of the `character_t`, with sub-types `pc_t` and `npc_t`. If you are using my code, you should change these so that they use C++ class inheritance (`pc` and `npc` should inherit from `character`). If you're not using my code, but you have a similar design, you should also use inheritance.

To the PC class we're going to add yet another map of the dungeon, this one will contain terrain—like the first map we ever made—however, it will be only the terrain that the PC can see or remembers having seen. The dungeon is now dark. In a later assignment, we'll implement objects, including lights, but for now we treat the PC as if it is carrying a light with a light radius of 3 dungeon cells (at most a 5×5 region of the dungeon will be illuminated). Everything within that radius is illuminated and thus visible. Dungeon terrain, once seen, is remembered.

Rendering is changed to render only visible monsters, and visible and remembered terrain. All unseen and unremembered terrain is rendered with spaces (like rock). Note that monsters can alter the dungeon. If the PC doesn't see this happening, what has been remembered doesn't change. For instance, dungeon cell (y, x) is rock and is in sight of the PC. The PC moves to a position where (y, x) is no longer visible, and a monster tunnels through that position, converting it to corridor. The PC hasn't witnessed that event and doesn't know about it. The rendered view still displays the cell as rock. The PC moves back so that (y, x) re-enters view and the remembered terrain is updated so that the cell is displayed as corridor.

We're also going to add some debugging commands. These are the kinds of commands that are useful to developers but don't make it into released games (sometimes they do make it in and get referred to as *cheats* by the game's player community). We'll add two such commands:

- Add support to display the dungeon without the fog of war (i.e., as it is currently displayed). This may (as you choose) be either a togglable effect (you turn it on, and later turn it off again) or a one turn effect (you display the fog-free dungeon, and after you make a move you display the foggy dungeon). In either case, map the functionality to the `f` key.
- Add support to teleport the PC. The `t` key will activate teleport mode. When in teleport mode, the movement keys (same as for moving the PC) will move a targeting pointer (for instance, `*`). A second `t` will then send the PC to the targeted location. `t` followed by `r` (while in targeting mode) will send the PC to a random location. Teleporting into rock is allowed (but not into immutable rock), but if the PC ends up surrounded by rock, it will not be able to move (except to teleport back out or wait for a monster to create a tunnel). The PC can *step out* of the rock onto adjacent floor, if available. It would be useful, but not required, to display the map without fog of war while targeting.

All new code should be written in C++.