COM S 327, Spring 2019
Programming Project 1.05
User Interface

Last week we added some characters, and made them move around. You may have added some code to drive your `@`; You can rip that code out, now[1]. We're going to add a user interface that you can use to drive your `@` manually. If you like, you can leave the auto-drive code in there and add a command to turn it on and off at runtime.

Still working in C, link in the ncurses library and use it for unbuffered I/O.

We're going to make our map exits, too. When the PC steps on an exit, instead of appearing on the exit, the PC appears on the road adjacent to the exit on the neighboring map. NPCs still cannot use exits.

How you choose to deal with your turn queue when moving to a new map is up to you. In my opinion, it is both easiest and most sensible to have maps remain static when the PC is not there, so if you leave a map and come back, the map will appear exactly as it did when you left. The easiest way to achieve this is probably to have a turn queue per map; however, this leads to a notion of a *map time* which is separate and distinct from *game time*. There's nothing strictly "wrong" with this, but it does feel a bit off to me, so I think I will work on a way to depopulate and repopluate my heap when moving such that I have a universal game time. This really only matters if we implement things which have an absolute time limit and should happen even if the PC is not on the map, for instance, the ripening of berries or apricorns.

As for user input, all commands are to be activated immediately upon key-press. There is never a need to hit enter. Any command which is not explicitly defined is a no-op. Implement the following commands:

Key(s)	Action
`7` or `y`	Attempt to move PC one cell to the upper left.
`8` or `k`	Attempt to move PC one cell up.
`9` or `u`	Attempt to move PC one cell to the upper right.
`6` or `l`	Attempt to move PC one cell to the right.
`3` or `n`	Attempt to move PC one cell to the lower right.
`2` or `j`	Attempt to move PC one cell down.
`1` or `b`	Attempt to move PC one cell to the lower left.
`4` or `h`	Attempt to move PC one cell to the left.
>	Attempt to enter a Pokémart or Pokémon Center. Works only if standing on a building. Leads to a user interface for the appropriate build. You may simply add a placeholder for this for now, which you exit with a <.
`5` or space or .	Rest for a turn. NPCs still move.
`t`	Display a list of trainers on the map, with their symbol and position relative to the PC (e.g.: "r, 2 north and 14 west").
up arrow	When displaying trainer list, if entire list does not fit in screen and not currently at top of list, scroll list up.
down arrow	When displaying trainer list, if entire list does not fit in screen and not currently at bottom of list, scroll list down.
escape	When displaying trainer list, return to character control.
`Q`	Quit the game. Your main game loop will become: `while (!quit_game) { ... }`

[1]If you gave your PC any special powers, like the ability to move through mountains or forrests, those should no longer apply, either; your PC is a normal human, save that he or she may be abnormally foolish.

If the player attempts to move to a map cell with an NPC, or if an NPC attempts to move to the PC's map cell, and the PC has not already defeated this trainer in a battle, a *Pokémon Battle* interface is triggered. For now, this interface is a placeholder. The only command in the interface is `escape`, to leave the battle, which marks the NPC as having been defeated. A defeated hiker or rival will no longer path to the PC (it's up to you how you would like them to move from this time on). If the PC attempts to move into the cell of a defeated trainer, nothing happens.

With these changes, we no longer need the delay that we built in last week; the game will now pause automatically for input. And ncurses should handle the redrawing, so we're no longer spewing the entire map to the terminal each turn. Things will look much nicer.

Note that the keys `y`, `k`, `u`, `l`, `n`, `j`, `b`, and `h` are not as strange or arbitrary as they may initially appear. vi and vim users will immediately recognize these as the cursor movement keys in their editor. They're also used in many roguelike games (including the original Rogue and most of its direct descendants).

Our maps fill 21 out of 24 lines in a terminal. Display them on lines 1–21 (zero indexed). The top line is for message display. Use it to display any messages you like (like debugging information, or information about why a command can't be executed ("There's a tree in the way!" or "A wild Riakou appears!")). The bottom 2 lines are for status information, which we'll deal with in a later assignment.

You may add any other commands if you like, or map the required commands to additional keys, as long as you implement the specified mappings. As this game is not yet fleshed out through the end of the semester, I cannot give you a list of reserved keystrokes, so you may be forced to change keys for any extra commands in future assignments.

Color!

Curses makes it easy to implement color. It is not required, but I highly recommend you color your terrain now. We'll talk about how to do this (and other Curses stuff) in lecture. If you already have color using Curses, the rest of this should be really easy for you. If you have color implemented some other way, you're going to have to change it to use Curses color.