COM S 327, Spring 2019
Programming Project 1.06
CSV parsing 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).

The major structs in your game should be changed to classes; this includes the map and the character structures. There are no requirements about access to members (public, private, etc.); you decide what you think is best. My code uses a pseudo-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`[1]). If you're not using my code, but you have a similar design, you should also use inheritance[2].

You do not have to do anything with `heap.h` and `heap.c`. These already include the necessary syntax to make them play nice with C++ and may remain as C files.

We'll be parsing in a number of data files that describe pokémon and their moves. The database containing these files is available on Pyrite under `/share/cs327/pokedex`. You may copy this entire database to a local location if you would like using scp, for example:

```
scp -r netid@pyrite.cs.iastate.edu:/share/cs327/pokedex /local/path/
```

where `netid` is your net id and `/local/path/` is the local path to the location where you want your files.

When loading the database, your program should look in at least two and optionally three places for the files, only failing if none of those locations contain the database. Your program should first look under `/share/cs327`. Failing that, it should look under `$HOME/.poke327/`. And, optionally, if may look in a third place of your choosing. For the second location, use `getenv()` to resolve the value of the `HOME` environment variable.

Your program should open the database in the first location where it is found, or print an error message and terminate upon failure.

The CSV files are located under `pokedex/pokedex/data/csv/`. We will need to parse the following files:

- `pokemon.csv`
- `moves.csv`
- `pokemon_moves.csv`
- `pokemon_species.csv`
- `experience.csv`
- `type_names.csv`

For each parsed file, create a struct or class to hold the data type, then create an array of those to hold all of the data.

Use the value -1 as a placeholder for empty cells in the CSV files.

[1] This is a requirement, not an optional change.

[2] Requiring this would be harder to enforce—a lot of manual effort for the TAs—so we won't do that, but you should make an effort to use inheritance somewhere in your code if the organization allows it, just so that you get the experience.

`type_names.csv` contains some non-ascii characters. You are welcome to fully parse the file—and to use non-English words—but if you are only interested in English, you can simple read whole lines and discard the irrelevant lines from this file. Only those lines wherein `local_language_id` is 9 are in English.

Modify your game to take a single command line parameter, one of `pokemon`, `moves`, `pokemon_moves`, `pokemon_species`, `experience`, or `type_names`. Your program should parse the specified file, print its contents to standard output (do not initialize curses), and exit. The whole game will still be compiled and linked, we're simply exiting before running it.

The format of the files is multiple lines of values separated by commas. The first line in each file defines the fields in the file (and make good choices for field names in your data structures). Some fields will be empty, denoted by a comma followed immediately by a comma. In printing out your parsed data, do not print the -1 that you use internally to denote empty fields.

All new code should be written in C++.