CPS506 - Comparative Programming Languages - Fall 2017

Assignment 4 - Rust

This assignment is a relatively simple program to capture various aspects of programming languages. This version is in Rust

The application is a simple Snakes and Ladders game described here. In addition to the specifications there, the following Elixir-specific parameters will apply:

Frequently Asked Questions
Course Management Form
Resources
Assignments
Tweet
Follow @RyersonCPS506

1. Do `cargo new assign4 --bin` to create your package. Save this `.gitignore` file in that directory.
2. Your module must have a `read_from` function that accepts a string.
3. Your module must have a `format_game` function that formats the current state of the board as a string.
4. When your `main` program is run, it must read commands from standard input, passing each line to the `readFrom` function. At the end of the input, it must print the state of the board on standard output.
5. Put your ownership information (see the assignment page) in the `assign4/README.md` file.
6. The CPS506 gitlab repo has sample `main.rs` and `testing.rs` files.
7. The marker should be able to run your program by entering the following code:

```
cargo run
board 3 4
players 2
turns 5
^D
```

You should do your assignment in the your Git `CPS506` repository in a folder called `assign4` (created with the `cargo` command above). Every time you have completed a part of the assignment, you should commit it to the repository. You shouldn't wait until everything is complete to do this, it's better to check in regularly. Remember to do `git status`, `git commit`, and `git commit` from somewhere within the repository periodically to make sure you're commiting all of your code. Also remember to **not** add binary files or other files that can be generated from the source. Create a `.gitignore` file that excludes commonly created files that should be excluded; add to that file if you notice any undesirable files being staged for commiting to the repository. You can so a `git add .` as many times as you want, but you only have to do it once each time there are new files to be included in the repository. In a terminal/command window simply change to the working directory and check-in, for example:

```
cd gitlab/cps506
git commit -m "finished code and tests for snakes and ladders"
git push
```

Dave Mason

Ryerson University