CS 4500 Project

Phase 3: Referee & Players, the Rule Checker, Game Board

Due: Wednesday, October 30, 11:59pm

Submission: Place the artifacts in your repository as follows (all paths are relative to the top-level of the project directory):

For the *Design Task*: place `referee.md` and `player.md` in the `Planning/` directory.

For the *Programming Task*: place `rules.PP` (or `Rules.PP`) in the `Common/` directory.

For the *Testing Task*: place `xboard` and `board-tests/` in a new directory called `3/`.

Extra information: To help graders evaluate your submission, you can include extra information in a file called `README.md` in the directory `3/`.

Design Task

A Tsuro game is an interaction between a component representing the game system (referee or game manager), and player components. Describe interfaces for these components.

Submit the design as two separate documents, each up to two pages long.

Programming Task

Implement the rule checking component.

In addition to the basic constraints of Tsuro (see Phase 1), for now, the following rules should be enforced by the rule checker, based on the state of the game:

> An initial placement of an avatar on a specific port p and tile at a coordinate (x, y) on the board is legal if the squares horizontally and vertically neighboring (x, y) are unoccupied and the port p is on a path that connects the edge of the board to an unoccupied square. *Note: although placements that violate this rule are rejected by the board itself (because they result in an invalid board state), the rule checker should allow checking for this as a service to the referee (game manager) and, potentially, players.*
>
> When a player requests the placement of a tile during its turn, the tile placement may not cause the player's suicide (as in exiting the board), **unless** this is the only possible option, based on the supplied tiles.

Everything else is legal.

Testing Task

Develop an external test harness for the game board.

Input: A JSON array describing the state of the game. The array contains initial and intermediate tile placements. An initial placement has the following shape:

```
[tile-index, rotation, color, port, x, y]
```

The initial placement contains:

- `tile-index` – from the range `0...34` – index of the tile
- `rotation` – one of `0`, `90`, `180`, `270` – the rotation of the tile
- `color` – one of `"white"`, `"black"`, `"red"`, `"green"`, `"blue"` – the color of the avatar on the tile
- `port` – from the range `"A"..."H"` – port where the avatar is placed
- `x` and `y` – both from the range `0...9` – the grid position of the tile

An intermediate placement has the shape

```
[color, tile-index, rotation, x, y]
```

where the fields are the same as above, but `color` refers to the player that requested the tile placement.

If the input is invalid, that is, if it does not follow the above format, or if it is rejected by the board, the test harness should exit showing an error message.

Output should be a JSON array of responses, one for each of the colors, where a response is one of the following:

- `[color, tile-index, rotation, port, x, y]` – the position of the avatar resulting from the given placements
- `[color, " never played"]` – if the avatar was not part of the initial state
- `[color, " exited"]` – if any of the tile placements results in the avatar exiting the game
- `[color, " collided"]` – if two or more avatars ended up in the same spot, that is if their paths connected.

Create five tests and place them in a new directory `3/board-tests/`. Include both inputs and expected outputs, named `n-in.json` and `n-out.json` (for test number `n`).