Assignment 4: Maze Game REST API

1. General

- All ID's are assigned by the back-end and are implementation specific. Therefore, you should not worry about trying to match the IDs that appear in the sample URLs below. The UI will adjust to work with whatever IDs you pass it.
- All commands return HTTP 200 (OK) unless stated otherwise.
- Any end-point accepting an ID (be it in the path, query string, or in the body) must return an HTTP 404 error if the ID does not exist.

`GET /api/about`

- Return your name as a string.

2. Games

`GET /api/games`

- Return a list of `ApiGameWrapper` objects for the games that the system knows about. Note that once a game is created, the server does not need to ever delete that game: it can exist until the server is restarted.
- Each `ApiGameWrapper` stores an ID to uniquely identify the game. This ID is assigned by the backend.
  - *TIP: Make the ID the game's index into the list which you use to store the games.*

`POST /api/games`

- Create a new game by POSTing to this end point. You need not post any data (no body).
- Returns one fully populated `ApiGameWrapper` object instance.
- Returns HTTP status code 201 (Created) when successful.

`GET /api/games/1`

- Return the `ApiGameWrapper` object for game 1 (where 1 is the game ID; 1 need not be the first ID assigned).
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.

3. Board

`GET /api/games/1/board`

- Return the current state of the board as one `ApiBoardWrapper` object (in JSON) for game 1 (change to the ID you need).
- Object contains:
  - board size
  - maze info
    - The 2D arrays for walls and visibility are accessible via `hasWall[row][col]`
    - The `isVisible` 2D array controls what cells in the board are currently visible to the user
  - cat/mouse/cheese placement as `ApiLocationWrapper` objects
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.

Moves

`POST /api/games/1/moves`

- Make a move in game 1 (change to the ID you need).
- Post one of the following strings, as the body of your HTTP POST request, to make a move:
  - “MOVE_UP”, “MOVE_DOWN”, “MOVE_LEFT”, “MOVE_RIGHT” move the mouse one square in that direction.
  - “MOVE_CATS”: have the game move all cats.
- No data is returned by this request (empty body): client expected to query other endpoints to find out board state after a move has been made.
- Returns HTTP status code 202 (Accepted) when successful.
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.
  - Return 400 (Bad Request) if the body of the message is not one of the above strings.
  - Return 400 (Bad Request) if the user has asked the mouse to move into a wall.

Cheats

`POST /api/games/1/cheatstate`

- Activate a cheat state for the game.
- Body of POST message must be one of:
  - “1_CHEESE”: Change the winning condition for the game to be 1 cheese.
  - “SHOW_ALL”: Permanently change the game to make all maze cells visible for this game.
- Returns HTTP status code 202 (Accepted) when successful.
- Error Handling:
  - Return 404 (File Not Found) if the requested game does not exist.
  - Return 400 (Bad Request) if the body of the message is not one of the above options.