BK TECHOUSE CODING EXERCISE

Exercise 1: Tic-tac-toe API

You will make an API that plays the game called **tic-tac-toe** in the US, and called naughts and crosses in some countries. Instructions for how to play the game are here if you've never played before:

http://www.exploratorium.edu/brain_explorer/tictactoe.html

Specifications

We will run an automated test suite against the URL you supply, so please stick to the specifications as best you can.

- Your server will be provided the current board in a GET request, using the 'board' parameter in the query string.
- If the board string doesn't represent a valid tic-tac-toe board, or it's not plausibly o's turn, your server should return an HTTP response code 400 (Bad Request)
- Your server always plays as o.
- Either player can go first.
- If the board is a valid tic-tac-toe board and it is plausibly o's turn, your server should return a string representation of the same board with one 'o' added.
- If possible and time permits, your tic-tac-toe api should play optimally (i.e. never lose when it is possible to force a tie, or tie when it is possible to win)
  - The best strategy is probably to search the game tree for winning moves
  - Here's a case-by-case analysis of what to do, with a bent towards beating a human: https://www.quora.com/Is-there-a-way-to-never-lose-at-Tic-Tac-Toe
  - a (possibly?) useful xkcd: https://xkcd.com/832/

Board representation

- The board is encoded as a string of nine characters where each character is either ‘o’ (letter o), ‘x’, or a space. The nine characters are the tic-tac-toe board read left to right, top to bottom -- for example:

```
x|o|
-+-+-
o|  |
-+-+-
 |x|
```

would be encoded with the string "xo o x ", and an empty board would be a string of nine spaces.

Example

If I run curl YOUR_URL?board=+xxo++o++

I should get the exact string oxxo o (that’s o-x-x-o-space-space-o-space-space) as the **entire** contents of the HTTP response body. If your api returns anything else, our unit tests will fail when run on your code.

Deploying

You can deploy anywhere on the public Internet that's convenient for you. We want to be able to test your code by hitting an API, so whatever way makes most sense for you to make that possible works for us. If you're not sure where to deploy here are a couple of good choices:

- Heroku is a solid choice that should get you up and running quickly and for free, and has plenty of documentation on how to get started (https://devcenter.heroku.com/start), and sample projects which you can clone and modify (feel free to use these!) If you haven't used Heroku before, we recommend creating a Heroku account and reading the getting-started instructions before you start the timed portion of the challenge.

- You can run the code locally on the computer you developed it on, and set up an ngrok tunnel (https://ngrok.com/product). You will run your code locally, and then run the ngrok program on your computer, and configure ngrok to forward requests to your public facing URL to your computer. This only works if your computer will be plugged in and running before our conversation so we can test!

Return to us

- A copy of your code (a github or gitlab link is fine)
- The url at which we can test it

Values

- You can use whatever libraries and tools you like, except please don't use a library that actually represents a Tic-Tac-Toe board or plays Tic-Tac-Toe -- that part is for you to design and build. Similarly, feel free to use google for general programming / library usage questions, but don't use code from the Internet which solves the game.
- What matters most is that your api works, i.e. that it implements the expected interface and I can play tic-tac-toe with it.
- In the same vein, it is more important that your API always return a valid board than that it use the optimal tic-tac-toe strategy.
- You should write code that you would want to work with and maintain in the future. We don't expect everything to be perfect, but if we can't read the code and understand what's going on, that's a problem.

Advice

- Use JavaScript ONLY.
- Deploy early and often; it will be much harder for us to judge your work if we can't hit your API to run it
- Pay attention to the API details - because we're running automated tests against the API, we can move onto debugging more interesting problems more quickly if we don't have to adapt our test suite to your solution.

Exercise 2: Simple job application system

System process flow

- Consider that an applicant wants to submit his / her profile and CV
- The HR is able to view the list of applicants
- That list should show only the first 10 items
- And that list should be sorted alphabetically
- The HR should be able to select an applicant and view his/her details
- Last, the HR must have option to change the application status (Dropped, Passed)

Required tools

Back-end : JavaScript Frameworks

Front-end : JavaScript Frameworks

Return to us

- A copy of your code (a github or gitlab link is fine)
- The url at which we can test it (Use same deployment process of your choice)