Palindrome Lab

- Due No Due Date
- Points 1
- Submitting a website url

(https://github.com/learn-co-curriculum/phase-1-algorithms-palindrome) (https://github.com/learn-co-curriculum/phase-1-algorithms-palindrome/issues/new)

Learning Goals

- Practice algorithmic problem solving

Introduction

Now that we've discussed what an algorithm is and defined a problem-solving approach, it's time to apply that to your first algorithm problem! Fork and clone this lab, then **read the instructions before running any tests**.

Instructions

Write a function `isPalindrome` that will receive one argument, a string. Your function should return `true` if the string is a palindrome (that is, if it reads the same forwards and backwards, like `"mom"` or `"racecar"`), and return `false` if it is not a palindrome.

To keep things simple, your function only needs to deal with lowercase strings that are all letters (don't worry about spaces or special characters).

Here are a few examples:

```
Input: "madam"
Output: true

Input: "robot"
Output: false
```

Problem Solving Approach

Use the **problem solving process** (https://github.com/learn-co-curriculum/phase-1-algorithms-what-is-an-algorithm) described in the previous lesson to come up with an approach to the problem and write your solution:

1. Rewrite the Problem in Your Own Words
2. Write Your Own Test Cases
3. Pseudocode
4. Code
5. Make It Clean and Readable
6. Optimize

Code your solution in the `index.js` file. There's space in the `index.js` file to write your pseudocode, but you're welcome to write pseudocode with pen and paper if you find that more beneficial.

You should also write some additional test cases in the `index.js` file to check your solution. After writing your test cases and coding your solution, you can view the output of your tests by running `node index.js` in the terminal.

For this exercise, you can use Google to look up syntax and any methods you might need to help solve this problem (note that not all interviewers will allow candidates to look up syntax, but some will).

When you're satisfied with your code, run `npm test` to run the tests and submit the lab using CodeGrade.

Conclusion

In the next two lessons, we'll review some solutions to this problem and talk through how to apply the problem solving process to this exercise. Take your time with this, and give it your best effort before reviewing the solution — it's ok if you don't end up with a working solution on the first try, so long as you make a deliberate effort at developing your problem solving process!