

## ✓ Coding Problems

### Objective

This assignment aims to demonstrate how to study a data structures or algorithms question in depth to prepare for an industry coding interview. Leetcode is a popular coding practice site that many use to practice for technical interviews. Like behavioral interviews, it's important to practice and keep your skills sharp.

### Group Size

Please complete this individually.

### Parts:

- Part 1: Figure out the problem you have been assigned, and understand the problem
- Part 2: Answer the questions about your assigned problem (including solving it)

### Part 1:

*You will be assigned one of three problems based of your first name. Enter your first name, in all lower case, execute the code below, and that will tell you your assigned problem. Include the output as part of your submission (do not clear the output). The problems are based-off problems from Leetcode.*

```
import hashlib

def hash_to_range(input_string: str) -> int:
    hash_object = hashlib.sha256(input_string.encode())
    hash_int = int(hash_object.hexdigest(), 16)
    return (hash_int % 3) + 1
input_string = "pulkit"
result = hash_to_range(input_string)
print(result)
```

↔ 2

## ✓ Question One: First Duplicate in List

### Description

Given a list of integers, return the **first value that appears more than once**. If there are multiple duplicates, return the one that appears **first** in the list. If no duplicate exists, return **-1**.

### Examples

Input: nums = [3, 1, 4, 2, 5, 1, 6]  
Output: 1

Input: nums = [7, 8, 9, 10]  
Output: -1

Input: nums = [4, 5, 6, 4, 6]  
Output: 4

### Question 1 Starter Code

```
from typing import List

def first_duplicate(nums: List[int]) -> int:
    # TODO
    pass
```

## ✓ Question Two: Valid Bracket Sequence

### Description

Given a string containing only the characters '(', ')', '{', '}', '[', and ']', determine if the input string is a **valid bracket sequence**.

A string is valid if:

- Open brackets are closed by the same type of brackets, and
- Open brackets are closed in the correct order.

### Examples

Input: s = "([{}])"  
Output: True

Input: s = "([)]"  
Output: False

Input: s = "()[]{}"  
Output: True

Input: s = "[{}]"  
Output: False

### Question 2 Starter Code

```
def is_valid_brackets(s: str) -> bool:
    # TODO
    pass
```

## ✓ Question Three: Move All Zeros to End

### Description

Given a list of integers, move all zeros to the end while maintaining the relative order of the non-zero elements.

### Examples

Input: nums = [0, 1, 0, 3, 12]  
Output: [1, 3, 12, 0, 0]

Input: nums = [4, 0, 5, 0, 0, 6]  
Output: [4, 5, 6, 0, 0, 0]

```
from typing import List
```

```
def move_zeros_to_end(nums: List[int]) -> List[int]:
    # TODO
    pass
```

## ✓ Part 2:

- Paraphrase the problem in your own words

**ANSWER:** Find that the string is valid, and contains () or {} or [] and return True, if all starting braces have matching closing braces, in correct order, then it is fine. So, set of braces, which are closed, if they have braces not getting closed inside, then they the function will return false.

- In this .ipynb file, there are examples that illustrate how the code should work (the examples provided above). Create 2 new examples for the question you have been assigned, that demonstrate you understand the problem. For question 1 and 2, you don't need to create the tree demonstration, just the input and output.

#### Answer

```
input_one = "([{}]" output_one = True
input_two = "{(){}]" output_two = False
```

- Code the solution to your assigned problem in Python (code chunk). Note: each problem can be solved more simply if you use an abstract data type that is suitable for that problem. Using that try to find the best time and space complexity solution!

```
def is_valid_brackets(s: str) -> bool:
    stack = []
    brackets = {'(': ')', '[': ']', '{': '}'

    for char in s:
        if char in '({[':
            stack.append(char)
        elif char in ')}]':
            if stack == [] or len(stack) == 0 or stack.pop() != brackets[char]:
                return False
        else:
            return False

    return len(stack) == 0

print("expected: False, returned:", is_valid_brackets("a"))
print("expected: False, returned:", is_valid_brackets("aa"))
print("expected: True, returned:", is_valid_brackets("{}[]{}"))
print("expected: True, returned:", is_valid_brackets("{}"))
print("expected: True, returned:", is_valid_brackets("()"))
print("expected: True, returned:", is_valid_brackets("[{}])")
print("expected: True, returned:", is_valid_brackets("{}[]{}"))
print("expected: True, returned:", is_valid_brackets("{}"))
print("expected: False, returned:", is_valid_brackets("(aa)"))
print("expected: False, returned:", is_valid_brackets("{}123}"))
print("expected: False, returned:", is_valid_brackets("{}123"))
print("expected: True, returned:", is_valid_brackets("{}{()}"))

print("expected: True, returned:", is_valid_brackets("{}{}{}"))
print("expected: False, returned:", is_valid_brackets("{}{}))"))
print("expected: True, returned:", is_valid_brackets("{}(){}{}"))
print("expected: False, returned:", is_valid_brackets("{}{}{}"))
```

```
↩ expected: False, returned: False
expected: False, returned: False
expected: True, returned: True
expected: True, returned: True
expected: True, returned: True
expected: True, returned: True
expected: True, returned: True
expected: True, returned: True
expected: False, returned: False
expected: False, returned: False
expected: False, returned: False
expected: True, returned: True
expected: True, returned: True
expected: False, returned: False
expected: True, returned: True
expected: False, returned: False
```

- Explain why your solution works

#### Answer

The solution focuses on making sure, the brackets are closing and opening in correct order. As we use Stack's ability to push and pop items, when new item is added is added to the end of the list, and if we encounter closing brackets, we pop the stack, which returns the last value of the stack, and if it is not the exact opposite of the

current position bracket, we return false, unless we reach the end of line, in which case, the bracket sequences are fine, so we return True.

- Explain the problem's time and space complexity

#### Answer

Time complexity:  $O(n)$  as we need to iterate through the loop.

Space complexity:  $O(n)$  as in case of worst case, we need to read the entire length.

- Explain the thinking to an alternative solution (no coding required, but a classmate reading this should be able to code it up based off your text)



#### Answer

Initially I was using the counter method, where I kept incremented counter int values for each opening bracket type `{ [ (`, and decrement the counter int value for each closing bracket type `} ] )`. But, it fails to account for the correct order and nesting. This question might be possible to do via recursion.

## Evaluation Criteria

- Problem is accurately stated
- Two examples are correct and easily understandable
- Correctness, time, and space complexity of the coding solution
- Clarity in explaining why the solution works, its time and space complexity
- Clarity in the proposal to the alternative solution

## Submission Information

 Please review our [Assignment Submission Guide](#)  for detailed instructions on how to format, branch, and submit your work. Following these guidelines is crucial for your submissions to be evaluated correctly.

### Submission Parameters:

- Submission Due Date: HH:MM AM/PM – DD/MM/YYYY
- The branch name for your repo should be: `assignment-1`
- What to submit for this assignment:
  - This Jupyter Notebook (`assignment_1.ipynb`) should be populated and should be the only change in your pull request.
- What the pull request link should look like for this assignment:  
`https://github.com/<your_github_username>/algorithms_and_data_structures/pull/<pr_id>`
  - Open a private window in your browser. Copy and paste the link to your pull request into the address bar. Make sure you can see your pull request properly. This helps the technical facilitator and learning support staff review your submission easily.

### Checklist:

- ☐ Create a branch called `assignment-1`.
- ☐ Ensure that the repository is public.
- ☐ Review [the PR description guidelines](#) and adhere to them.
- ☐ Verify that the link is accessible in a private browser window.

If you encounter any difficulties or have questions, please don't hesitate to reach out to our team via our Slack at `#cohort-3-help`. Our Technical Facilitators and Learning Support staff are here to help you navigate any challenges.

