MIDTERM CSTP-1305

There are 5 questions in total. If you are able to provide 2 solutions for any of the questions then you can skip 1 question. The only catch there is that the 2 solutions for that question should have different time complexity. I.e O(n*2) and O(n)

Additionally you have to provide time complexity for each solution you provide. You have to submit a git repository with the questions solved as you have been doing for assignments.

Question1: Find the missing number in a unsorted array whose value is upto n. For example if n is 5, the array will be
Let array = [5, 4,1,2] , here the **number missing is 3,** so you have to find that number. Note: there can be no negative number in the array, and the array always starts with 1.
Another example - If n is 10 ,
Let array = [9, 5, 3, 2,6, 1, 7, 8, 10], here the **missing number is 4.**

```
Write a function findMissingNumber(array, n) {
    // Returns missing number
}
```

TEST CASES

```
array = [5, 4, 1, 2]
n = 5
# Expected output: 3

array = [9, 5, 3, 2, 6, 1, 7, 8, 10]
n = 10
# Expected output: 4

array = [2, 3, 1, 5]
n = 5
# Expected output: 4

array = [1, 2, 3, 4, 5]
n = 6
# Expected output: 6
```

Question2: Given an array of integers `nums` and an integer `target`, return the indices of the two numbers such that they add up to the target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

For example - Let array = [1, 5, 2, 7] , target = 8 , so you have to **return [0, 3]**

Another example - Let array = [20, 1, 5, 2, 11], target = 3, so you have to **return [1, 3]**

TEST CASES

array = [1, 5, 2, 7]
target = 8
# Expected output: [1, 3]

array = [20, 1, 5, 2, 11]
target = 3
# Expected output: [1, 3]

array = [3, 2, 4]
target = 6

Expected output: [1, 2]

Question3: Generate All Permutations of a String

Given a string `str`, write a recursive function to generate all permutations of the string.

// Input: "ABC"
// Output: ["ABC", "ACB", "BAC", "BCA", "CAB", "CBA"]

Other example
// Input: "AB"
// Output: ["AB", "BA"]

Write a function generatePermutation(str) {
// Return array of combinations
}

TEST CASES:

```
let input = "AB";
// Expected Output: ["AB", "BA"]

let input = "A";
// Expected Output: ["A"]
```

Let input = “ABC”;
// Expected Output: ["ABC", "ACB", "BAC", "BCA", "CAB", "CBA"]
“

Question4: Given a linked list, determine if it has a cycle in it. If it has a cycle return true otherwise return false. In this solution you have to write a function like

Function checkIfCycleExists(headNode) {
 // Return true or false.
}

Note: You can create a linked list by yourself just like we did in class, and then pass the head node for testing your solution.

TEST CASES:
headNode = "A";
Linked List = “A” -> “B” -> “C” -> “A”
// Expected Output: true

headNode = "1";
Linked List = “1” -> “2” -> “3” -> null
// Expected Output: false

headNode = "1";
Linked List = “1” -> “2” -> “3” -> 1
// Expected Output: true

Question5: Given a string containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

- **Open brackets must be closed by the same type of brackets.**
- **Open brackets must be closed in the correct order.**
- **An empty string is also considered valid.**

Write a function like checkIfValidParenthesis(str)

```
{
   // return true or false
}
```

TEST CASES:

str = "()";
// Expected Output: true

str = "(){}[ ]";
// Expected Output: true

Str = “([})”
// Expected Output: false

Str = “[({})]”
// Expected Output: true