Leet Code Assignment

Name: K Lalith Aditya

Regd No: 22231

Date: 31-10-2022

Question 1)

1. Two Sum

Easy 39542 1274 Add to List Share

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

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

You can return the answer in any order.

Answer 1)

```python
#nums = [2,7,11,15]

#target = 9

class Solution:
    def twoSum(self,nums,target):
        for i in range(0,len(nums)):
            for j in range(i+1,len(nums)):
                if(j>len(nums)):
                    break
                local = nums[i] + nums[j];
                if(local == target):
                    return ([i,j])

#output = twosum(nums,target)
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 49 ms

Your input
```
[2,7,11,15]
9
```

Output
```
[0,1]
```

Diff

Expected
```
[0,1]
```

Question 2)

4. Median of Two Sorted Arrays

Hard 20392 2309 Add to List Share

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be `O(log (m+n))`.

Answer 2)

```
class Solution:
    def findMedianSortedArrays(self, nums1,nums2):
        for i in range(0,len(nums2)):
            nums1.append(nums2[i])
        # Bubble Sort
        for i in range(0,len(nums1)-1):
            for j in range(0,len(nums1)-i-1):
                if (nums1[j]>nums1[j+1]):
                    nums1[j], nums1[j+1] = nums1[j+1], nums1[j]
        #
        if((len(nums1)%2)!=0): # i.e. its odd
            a = nums1[floor(len(nums1)/2)]
            b = nums1[floor((len(nums1)/2)+1)]
            bob = int((a+b)/2)
            return bob
        else: # i.e. its even
            return int(nums1[len(nums1)/2])
```

Question 3)

9. Palindrome Number

Easy 7678 2340 Add to List Share

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Answer 3)

```python
class Solution:
    def isPalindrome(self,x):
        x = str(x)
        a = x[::-1]
        if(x == a):
            return True
        else:
            return False
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 55 ms

Your input	121
Output	true
Expected	true

Diff

Question 4)

20. Valid Parentheses

Easy 16418 836 Add to List Share

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

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Answer 4)

```
class Solution:
    def isValid(self, s):
        flag = 0
        for i in range(len(s)-1):
            if(i+1<=len(s)):
                if(s[i] == '(' and s[i+1]==')'):
                    flag = 0
                if(s[i] == '(' and s[i+1]!=')'):
                    flag = 1
                elif(s[i] == ')'):
                    flag = 1
                if(s[i] == '{' and s[i+1]=='}'):
                    flag = 0
                if(s[i] == '{' and s[i+1]!='}'):
                    flag = 1
                elif(s[i] == '}'):
                    flag = 1
                if(s[i] == '[' and s[i+1]==']'):
                    flag = 0
                if(s[i] == '[' and s[i+1]!=']'):
                    flag = 1
                elif(s[i] == ']'):
                    flag = 1
        if (flag == 1):
            return False
        else:
            return True
```

Question 5)

35. Search Insert Position

Easy 10376 486 Add to List Share

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with `O(log n)` runtime complexity.

Answer 5)

```
class Solution:
    def searchInsert(self,nums,target): # Binary Search
        low = 0
        high = len(nums)-1
        flag = 0
        while low <= high:
            mid = int((high + low)/2)
            # If x is greater, ignore left half
            if nums[mid] < target:
                low = mid + 1

            # If x is smaller, ignore right half
            elif nums[mid] > target:
                high = mid - 1

            # means x is present at mid
            else:
                flag = 1
                return mid

        if flag == 0 :
            i = 0
            while(i<len(nums)):
                if (target <nums[i]):
                    return i+1
                else:
                    continue
```

##########################