**CSA0674 - DAA ASSIGNMENT 1**

1. ***TWO SUM :***

**def two\_sum(nums, target):**

**num\_dict = {}**

**for i, num in enumerate(nums):**

**complement = target - num**

**if complement in num\_dict:**

**return [num\_dict[complement], i]**

**num\_dict[num] = i**

**print(two\_sum([2, 7, 11, 15], 9))**

**# Output: [0, 1]**

**print(two\_sum([3, 2, 4], 6))**

**# Output: [1, 2]**

**print(two\_sum([3, 3], 6))**

**# Output: [0, 1]**

1. ***ADD TWO NUMBERS :***

**class ListNode:**

**def \_\_init\_\_(self, val=0, next=None):**

**self.val = val**

**self.next = next**

**def add\_two\_numbers(l1, l2):**

**dummy = ListNode()**

**current = dummy**

**carry = 0**

**while l1 or l2 or carry:**

**val1 = l1.val if l1 else 0**

**val2 = l2.val if l2 else 0**

**carry, out = divmod(val1 + val2 + carry, 10)**

**current.next = ListNode(out)**

**current = current.next**

**l1 = l1.next if l1 else None**

**l2 = l2.next if l2 else None**

**return dummy.next**

**def create\_linked\_list(lst):**

**dummy = ListNode()**

**current = dummy**

**for number in lst:**

**current.next = ListNode(number)**

**current = current.next**

**return dummy.next**

**def print\_linked\_list(node):**

**while node:**

**print(node.val, end=" -> ")**

**node = node.next**

**print("None")**

**l1 = create\_linked\_list([2, 4, 3])**

**l2 = create\_linked\_list([5, 6, 4])**

**result = add\_two\_numbers(l1, l2)**

**print\_linked\_list(result)**

**# Output: 7 -> 0 -> 8 -> None**

1. ***LONGEST SUBSTRING WITHOUT REPEATING CHARACTERS :***

**def length\_of\_longest\_substring(s):**

**char\_index = {}**

**left = 0**

**max\_length = 0**

**for right, char in enumerate(s):**

**if char in char\_index and char\_index[char] >= left:**

**left = char\_index[char] + 1**

**char\_index[char] = right**

**max\_length = max(max\_length, right - left + 1)**

**return max\_length**

**print(length\_of\_longest\_substring("abcabcbb"))**

**# Output: 3**

**print(length\_of\_longest\_substring("bbbbb"))**

**# Output: 1**

**print(length\_of\_longest\_substring("pwwkew"))**

**# Output: 3**

1. ***MEDIAN OF TWO SORTED ARRAYS :***

**def find\_median\_sorted\_arrays(nums1, nums2):**

**nums = sorted(nums1 + nums2)**

**mid = len(nums) // 2**

**if len(nums) % 2 == 0:**

**return (nums[mid - 1] + nums[mid]) / 2**

**else:**

**return nums[mid]**

**print(find\_median\_sorted\_arrays([1, 3], [2]))**

**# Output: 2.0**

**print(find\_median\_sorted\_arrays([1, 2], [3, 4]))**

**# Output: 2.5**

1. ***LONGEST PALINDROMIC SUBSTRING :***

**def longest\_palindrome(s):**

**def expand\_around\_center(left, right):**

**while left >= 0 and right < len(s) and s[left] == s[right]:**

**left -= 1**

**right += 1**

**return s[left + 1:right]**

**result = ""**

**for i in range(len(s)):**

**odd\_palindrome = expand\_around\_center(i, i)**

**even\_palindrome = expand\_around\_center(i, i + 1)**

**result = max(result, odd\_palindrome, even\_palindrome, key=len)**

**return result**

**print(longest\_palindrome("babad"))**

**# Output: "bab" or "aba"**

**print(longest\_palindrome("cbbd"))**

**# Output: "bb"**

1. ***ZIGZAG CONVERSION :***

**def convert(s, numRows):**

**if numRows == 1 or numRows >= len(s):**

**return s**

**res = [''] \* numRows**

**index, step = 0, 1**

**for char in s:**

**res[index] += char**

**if index == 0:**

**step = 1**

**elif index == numRows - 1:**

**step = -1**

**index += step**

**return ''.join(res)**

**print(convert("PAYPALISHIRING", 3))**

**# Output: "PAHNAPLSIIGYIR"**

**print(convert("PAYPALISHIRING", 4))**

**# Output: "PINALSIGYAHRPI"**

1. ***REVERSE INTEGER :***

**def reverse(x):**

**sign = -1 if x < 0 else 1**

**x = abs(x)**

**rev = 0**

**while x != 0:**

**rev = rev \* 10 + x % 10**

**x //= 10**

**rev \*= sign**

**if rev < -2\*\*31 or rev > 2\*\*31 - 1:**

**return 0**

**return rev**

**print(reverse(123))**

**# Output: 321**

**print(reverse(-123))**

**# Output: -321**

**print(reverse(120))**

**# Output: 21**

1. ***STRING TO INTEGER (ATOI) :***

**def my\_atoi(s):**

**s = s.strip()**

**if not s:**

**return 0**

**sign = 1**

**start = 0**

**if s[0] in ['-', '+']:**

**sign = -1 if s[0] == '-' else 1**

**start = 1**

**result = 0**

**for i in range(start, len(s)):**

**if not s[i].isdigit():**

**break**

**result = result \* 10 + int(s[i])**

**result \*= sign**

**result = max(-2\*\*31, min(result, 2\*\*31 - 1))**

**return result**

**print(my\_atoi("42"))**

**# Output: 42**

**print(my\_atoi(" -42"))**

**# Output: -42**

**print(my\_atoi("4193 with words"))**

**# Output: 4193**

1. ***PALINDROME NUMBER :***

**def is\_palindrome(x):**

**if x < 0:**

**return False**

**return str(x) == str(x)[::-1]**

**print(is\_palindrome(121))**

**# Output: True**

**print(is\_palindrome(-121))**

**# Output: False**

**print(is\_palindrome(10))**

**# Output: False**

1. ***REGULAR EXPRESSION MATCHING :***

**def is\_match(s, p):**

**dp = [[False] \* (len(p) + 1) for \_ in range(len(s) + 1)]**

**dp[0][0] = True**

**for j in range(1, len(p) + 1):**

**if p[j - 1] == '\*':**

**dp[0][j] = dp[0][j - 2]**

**for i in range(1, len(s) + 1):**

**for j in range(1, len(p) + 1):**

**if p[j - 1] == '.' or p[j - 1] == s[i - 1]:**

**dp[i][j] = dp[i - 1][j - 1]**

**elif p[j - 1] == '\*':**

**dp[i][j] = dp[i][j - 2] or (dp[i - 1][j] if p[j - 2] == s[i - 1] or p[j - 2] == '.' else False)**

**return dp[-1][-1]**

**print(is\_match("aa", "a"))**

**# Output: False**

**print(is\_match("aa", "a\*"))**

**# Output: True**

**print(is\_match("ab", ".\*"))**

**# Output: True**