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]
```

2) *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
```

3) 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
```

4) *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
```

5) *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"
```

6) 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"
```

7) *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
```

8) *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
```

9) 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
```

10)   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
```