

## ASSIGNMENT-1

### 1.TWO SUM

```
def two_sum(n, target):  
    index = {}  
  
    for i, num in enumerate(n):  
        complement = target - num  
        if complement in index:  
            return [index[complement], i]  
        index[num] = i
```

```
n = [2, 7, 11, 15]  
target = 9  
print(two_sum(n,target))
```

### 2.ADD TWO NUMBERS

```
class ListNode:  
    def __init__(self, val=0, next=None):  
        self.val = val  
        self.next = next  
  
def addTwoNumbers(l1, l2):  
    dummy = ListNode(0)  
    current = dummy  
    carry = 0  
  
    while l1 or l2 or carry:  
        sum_val = (l1.val if l1 else 0) + (l2.val if l2 else 0) + carry  
        carry, val = divmod(sum_val, 10)  
        current.next = ListNode(val)  
        current = current.next  
        l1 = l1.next if l1 else None
```

```

l2 = l2.next if l2 else None

return dummy.next

l1 = ListNode(2, ListNode(4, ListNode(3)))
l2 = ListNode(5, ListNode(6, ListNode(4)))
result = addTwoNumbers(l1, l2)

while result:
    print(result.val, end=" ")
    result = result.next

```

### 3.LONGEST SUBSTRING

```

def longest_substring(s: str) -> int:
    char_index_map = {}
    start = max_length = 0
    for end, char in enumerate(s):
        if char in char_index_map and char_index_map[char] >= start:
            start = char_index_map[char] + 1
        char_index_map[char] = end
        max_length = max(max_length, end - start + 1)
    return max_length

s = "abcabcbb"
print(longest_substring(s))

```

### 4.MEDIAN OF TWO SORTED ARRAYS

```

def findMedianSortedArrays(n1, n2):
    nums = sorted(n1 + n2)
    n = len(nums)
    if n % 2 == 1:
        return nums[n // 2]

```

```

else:
    return (nums[n // 2 - 1] + nums[n // 2]) / 2.0
n1 = [1, 3]
n2 = [2]
print(findMedianSortedArrays(n1, n2))

```

## 5.LONGEST PALINDROME

```

def longest_palindromic_substring(s):
    def is_palindrome(s):
        return s == s[::-1]
    longest_palindrome = ""
    for i in range(len(s)):
        for j in range(i, len(s)):
            substring = s[i:j+1]
            if is_palindrome(substring) and len(substring) > len(longest_palindrome):
                longest_palindrome = substring
    return longest_palindrome
s = "babad"
print(longest_palindromic_substring(s))

```

## 6.ZIG-ZAG

```

def convert(s: str, numRows: int) -> str:
    if numRows == 1 or numRows >= len(s):
        return s
    rows = [""] * numRows
    row, step = 0, -1
    for char in s:
        rows[row] += char
        if row == 0 or row == numRows - 1:
            step = -step

```

```
        row += step
    return ''.join(rows)

input = "PAYPALISHIRING"

num_rows = 3

print(convert(input, num_rows))
```

## **7.REVERSE**

```
num=1234

rev=0

while num!=0:

    rem=num%10

    rev=rev*10+rem

    num//=10

print(rev)
```

## **8.CONVERT STRING TO INTEGER**

```
str="42"

print(int(str))
```

## **9.PALINDROME OR NOT**

```
num=127

temp=num

rev=0

while num>0:

    rem=num%10

    rev=rev*10+rem

    num=num//10

if temp==rev:
```

```
    print("palindrome")
else:
    print("not palindrome")
```

```
import re
```

## **10.REGULAR EXPRESSION MATCHING**

```
p = "aa"
```

```
s = "a"
```

```
p = r"{}".format(p)
```

```
p = re.compile(p)
```

```
if p.fullmatch(s):
```

```
    print("true")
```

```
else:
```

```
    print("false")
```