1.TOW SUM:

Program:

```
class Solution:
def twoSum(self, nums: List[int], target: int) -> List[int]:
hashmap = {}
for i in range(len(nums)):
complement = target - nums[i]
if complement in hashmap:
return [i, hashmap[complement]]
hashmap[nums[i]] = i
# Return an empty list if no solution is found
 return []
```

Output:

Input: nums = [3,2,4], target = 6

Output: [1,2]

2.ADD TWO NUMBERS:

Program:

```
class Solution:
def addTwoNumbers(self, l1: Optional[ListNode], l2: Optional[ListNode]) -> Optional[ListNode]:
dummy = ListNode()
res = dummy
total = carry = 0
while l1 or l2 or carry:
total = carry
if l1:
 total += l1.val
l1 = l1.next
 if l2:
total += l2.val
```

```
 l2 = l2.next
 num = total % 10
carry = total // 10
dummy.next = ListNode(num)
dummy = dummy.next
return res.next
```

Output:

Input: l1 = [0], l2 = [0]

Output: [0]

3.LONGEST SUBSTRING WITHOUT REAPETING CHARACTERS:

Program:

```
class Solution:
def lengthOfLongestSubstring(self, s: str) -> int:
left = max_length = 0
char_set = set()
for right in range(len(s)):
while s[right] in char_set:
char_set.remove(s[left])
left += 1
char_set.add(s[right])
max_length = max(max_length, right - left + 1)
return max_length
```

Output:

Input: s = "abcabcbb"

Output:

4.MEDIAN OF TWO SORTED ARRAYS:

Program:

```
class Solution:
def findMedianSortedArrays(
```

```
self, nums1: List[int], nums2: List[int]
) -> float:
m, n = len(nums1), len(nums2)
p1, p2 = 0, 0
# Get the smaller value between nums1[p1] and nums2[p2].
def get_min():
nonlocal p1, p2
if p1 < m and p2 < n:
if nums1[p1] < nums2[p2]:
ans = nums1[p1]
p1 += 1
else:
ans = nums2[p2]
p2 += 1
elif p2 == n:
ans = nums1[p1]
p1 += 1
else:
ans = nums2[p2]
p2 += 1
return ans
if (m + n) % 2 == 0:
for _ in range((m + n) // 2 - 1):
_ = get_min()
return (get_min() + get_min()) / 2
else:
for _ in range((m + n) // 2):
_ = get_min()
return get_min()
```

Output:

Input: nums1 = [1,3], nums2 = [2]

Output: 2.00000

5.LONGEST PALINDROMIC SUBSTRING:

Program:

```
class Solution:
def longestPalindrome(self, s: str) -> str:
def check(i, j):
left = i
right = j - 1
while left < right:
if s[left] != s[right]:
return False
left += 1
right -= 1
return True
for length in range(len(s), 0, -1):
for start in range(len(s) - length + 1):
if check(start, start + length):
return s[start : start + length]
return ""
```

Output:

Input: s = "babad"

Output: "bab"

6.ZIGZAG CONVERSION:

Program:

```
class Solution:
def convert(self, s: str, numRows: int) -> str:
if numRows == 1 or numRows >= len(s):
```

```
return s

idx, d = 0, 1

rows = [[] for _ in range(numRows)]

for char in s:

rows[idx].append(char)

if idx == 0:

d = 1

elif idx == numRows - 1:

d = -1

idx += d

for i in range(numRows):

rows[i] = ''.join(rows[i])

return ''.join(rows)
```

Output:

Input: s = "PAYPALISHIRING", numRows = 3

Output: "PAHNAPLSIIGYIR"

7.REVERSE INTEGER:

Program:

```
class Solution:

def reverse(self, x: int) -> int:

sign = [1, -1][x < 0]

rev, x = 0, abs(x)

while x:

x, mod = divmod(x, 10)

rev = rev * 10 + mod

if rev > 2**31 - 1:

return 0

return sign * rev
```

Output:

Input: x = 123

Output: 321

8.STRING TO INTEGER (ATOI):

Program:

```
class Solution:

def myAtoi(self, s: str) -> int:

if not s:

return 0

# Constants for 32-bit signed integer range

INT_MAX = 2**31 - 1

INT_MIN = -2**31

i = 0

n = len(s)

# Step 1: Skip leading whitespace

while i < n and s[i] == ' ':

i += 1

# Check if we've reached the end

if i == n:

return 0

# Step 2: Check for sign

sign = 1

if s[i] == '+':

i += 1

elif s[i] == '-':

sign = -1

i += 1

# Step 3: Read digits and convert

res = 0

while i < n and s[i].isdigit():
```

```
digit = int(s[i])

res = res * 10 + digit

if sign * res <= INT_MIN:

return INT_MIN

if sign * res >= INT_MAX:

return INT_MAX

i += 1

# Step 4: Apply sign and return

return res * sign
```

Output:

Input: s = "42"

Output: 42

9.PALINDROME NUMBER:

Program:

```
class Solution:

def isPalindrome(self, x: int) -> bool:

if x < 0:

return False

reverse = 0

xcopy = x

while x > 0:

reverse = (reverse * 10) + (x % 10)

x //= 10

return reverse == xcopy
```

Output:

Input: x = 121

Output: true

10.REGULAR EXPRESSION MATCHING:

Program:

```
class Solution(object):

def isMatch(self, text: str, pattern: str) -> bool:

if not pattern:

return not text

first_match = bool(text) and pattern[0] in {text[0], "."}

if len(pattern) >= 2 and pattern[1] == "*":

return (

self.isMatch(text, pattern[2:])

or first_match

and self.isMatch(text[1:], pattern)

)

else:

return first_match and self.isMatch(text[1:], pattern[1:])
```

Output:

Input: s = "aa", p = "a"

Output: false