1. **[Two Sum](https://leetcode.com/problems/two-sum/):-**

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

**Program:**

class Solution:

def twoSum(self, nums, target):

d = {}

for i,

n in enumerate(nums):

if target - n in d: return [d[target - n], i]

d[n] = i

1. [**Add Two Numbers**](https://leetcode.com/problems/add-two-numbers/):-

**Program statement**: given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

**Program**:

class Solution:

    def addTwoNumbers(self, l1, l2):

        dummy = curr = ListNode(0)

        carry = 0

        while l1 or l2 or carry:

            carry, val = divmod((l1.val if l1 else 0) + (l2.val if l2 else 0) + carry, 10)

            curr.next = ListNode(val)

            curr = curr.next

            l1 = l1.next if l1 else None

            l2 = l2.next if l2 else None

        return dummy.next

1. [**Longest Substring Without Repeating Characters**](https://leetcode.com/problems/longest-substring-without-repeating-characters/)**:-**

**Program statement:** Given a string s, find the length of the longest substring without duplicate characters.

**Program:**

class Solution(object):

    def lengthOfLongestSubstring(self, s):

        seen = {}

        left = max\_len = 0

        for right, char in enumerate(s):

            if char in seen and seen[char] >= left:

                left = seen[char] + 1

            seen[char] = right

            max\_len = max(max\_len, right - left + 1)

        return max\_len

1. [**Median of Two Sorted Arrays**](https://leetcode.com/problems/median-of-two-sorted-arrays/):-

**Program statement:**

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

**Program:**

class Solution:

    def findMedianSortedArrays(self, nums1, nums2):

        nums = sorted(nums1 + nums2); m = len(nums)

        return (nums[m//2] + nums[~(m//2)]) / 2.0

1. [**Longest Palindromic Substring**](https://leetcode.com/problems/longest-palindromic-substring/)**:-**

**Program statement:**

Given a string s, return *the longest* *palindromic* *substring* in s.

**Program:**

class Solution(object):

    def longestPalindrome(self, s):

        if not s:

            return ""

        start = 0

        max\_len = 1

        for i in range(len(s)):

            # Check odd length palindrome

            l, r = i, i

            while l >= 0 and r < len(s) and s[l] == s[r]:

                if r - l + 1 > max\_len:

                    start = l

                    max\_len = r - l + 1

                l -= 1

                r += 1

            # Check even length palindrome

            l, r = i, i + 1

            while l >= 0 and r < len(s) and s[l] == s[r]:

                if r - l + 1 > max\_len:

                    start = l

                    max\_len = r - l + 1

                l -= 1

                r += 1

        return s[start:start + max\_len]

[**6. Zigzag Conversion**](https://leetcode.com/problems/zigzag-conversion/):-

**Program statement:**

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

**Program:**

class Solution(object):

    def convert(self, s, numRows):

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

            return s

        rows = [''] \* numRows

        curr, step = 0, 1

        for c in s:

            rows[curr] += c

            if curr == 0: step = 1

            if curr == numRows - 1: step = -1

            curr += step

        return ''.join(rows)

1. [**Reverse Integer**](https://leetcode.com/problems/reverse-integer/)**:-**

**Program statement:** Given a signed 32-bit integer x, return x*with its digits reversed*. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0.

**Program:**

class Solution(object):

    def reverse(self, x):

        sign = -1 if x < 0 else 1

        x\_abs = abs(x)

        rev = 0

        while x\_abs:

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

            x\_abs //= 10

        rev \*= sign

        # Check 32-bit signed int overflow

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

            return 0

        return rev

1. [**String to Integer (atoi)**](https://leetcode.com/problems/string-to-integer-atoi/)**:-**

**Program statement:**

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer.

**Program:**

class Solution(object):

    def myAtoi(self, s):

        s = s.lstrip()  # remove leading spaces

        if not s:

            return 0

        sign = 1

        i = 0

        if s[0] in ('-', '+'):

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

            i += 1

        result = 0

        INT\_MAX, INT\_MIN = 2\*\*31 - 1, -2\*\*31

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

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

            i += 1

**9**[**. Palindrome Number**](https://leetcode.com/problems/palindrome-number/):-

**Program statement:**

**Program:**

class Solution(object):

    def isPalindrome(self, x):

        if x < 0 or (x % 10 == 0 and x != 0):

            return False

        rev = 0

        while x > rev:

            rev = rev \* 10 + x % 10

            x //= 10

        return x == rev or x == rev // 10

[**10. Regular Expression Matching**](https://leetcode.com/problems/regular-expression-matching/)**:-**

**Program statement:**

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*'

**Program:**

class Solution(object):

    def isMatch(self, s, p):

        memo = {}

        def dfs(i, j):

            if (i, j) in memo:

                return memo[(i, j)]

            if j == len(p):

                return i == len(s)

            first = i < len(s) and p[j] in {s[i], '.'}

            if j + 1 < len(p) and p[j + 1] == '\*':

                memo[(i, j)] = dfs(i, j + 2) or (first and dfs(i + 1, j))

            else:

                memo[(i, j)] = first and dfs(i + 1, j + 1)

            return memo[(i, j)]

        return dfs(0, 0)