1. How to find all permutations of String?
   Solution:

```
ini_str = "abc"
print("Initial string", ini_str)
result = []
def permute(data, i, length):
        if i == length:
                result.append(''.join(data) )
        else:
                for j in range(i, length):
                        data[i], data[j] = data[j], data[i]
                        permute(data, i + 1, length)
                        data[i], data[j] = data[j], data[i]
permute(list(ini_str), 0, len(ini_str))
print("Resultant permutations", str(result))
```

2. How to find the middle element of a singly linked list in one pass?
   Solution:

```
class Node:
        def __init__(self, data):
                self.data = data
                self.next = None
class LinkedList:
        def __init__(self):
                self.head = None
        def push(self, new_data):
                new_node = Node(new_data)
                new_node.next = self.head
                self.head = new_node
        def printMiddle(self):
                slow_ptr = self.head
                fast_ptr = self.head
                if self.head is not None:
                 while (fast_ptr is not None and fast_ptr.next is not None):
                                fast_ptr = fast_ptr.next.next
                                slow_ptr = slow_ptr.next
                        print("The middle element is: ", slow_ptr.data)
list1 = LinkedList()
```

```
list1.push(5)
list1.push(4)
list1.push(2)
list1.push(3)
list1.push(1)
list1.printMiddle()
```

3. How to reverse a linked list?
   Solution:

```
class Node:
        def __init__(self, data):
                self.data = data
                self.next = None
class LinkedList:
        def __init__(self):
                self.head = None
        def reverse(self):
                prev = None
                current = self.head
                while(current is not None):
                        next = current.next
                        current.next = prev
                        prev = current
                        current = next
                self.head = prev
        def push(self, new_data):
                new_node = Node(new_data)
                new_node.next = self.head
                self.head = new_node
        def printList(self):
                temp = self.head
                while(temp):
                        print (temp.data,end=" ")
                        temp = temp.next
llist = LinkedList()
llist.push(20)
llist.push(4)
llist.push(15)
llist.push(85)
```

```
print ("Given Linked List")
llist.printList()
llist.reverse()
print ("\nReversed Linked List")
llist.printList()
```

4. How to find the 3rd node from the end in a singly linked list?
   Solution:

```
class Node:
        def __init__(self, new_data):
                self.data = new_data
                self.next = None
class LinkedList:
        def __init__(self):
                self.head = None
        def push(self, new_data):
                new_node = Node(new_data)
                new_node.next = self.head
                self.head = new_node
        def printNthFromLast(self, n):
                temp = self.head # Used temp variable
                length = 0
                while temp is not None:
                        temp = temp.next
                        length += 1
                if n > length:
                   print('Location is greater than the' +' length of LinkedList')
                        return
                temp = self.head
                for i in range(0, length - n):
                        temp = temp.next
                print(temp.data)
if __name__ == "__main__":
        llist = LinkedList()
        llist.push(20)
        llist.push(4)
        llist.push(15)
        llist.push(35)
        llist.printNthFromLast(4)
```

5. How do you find the sum of two linked lists using Stack?
   Solution:

6. Write a program to locate the left insertion point for a specified value in sorted order
   Solution:

```
import bisect
def index(a, x):
   i = bisect.bisect_left(a, x)
  return i
a = [1,2,4,5]
print(index(a, 6))
print(index(a, 3))
```

7. Write a program to locate the right insertion point for a specified value in sorted order.
   Solution:

```
import bisect
def index(a, x):
   i = bisect.bisect_right(a, x)
  return i
a = [1,2,4,5]
print(index(a, 6))
print(index(a, 3))
```

8. Write a program to insert items into a list in sorted order.
   Solution:

```
def insert(list, n):
        index = len(list)
        for i in range(len(list)):
        if list[i] > n:
                index = i
                break
        if index == len(list):
```

```
        list = list[:index] + [n]
        else:
        list = list[:index] + [n] + list[index:]
        return list
list = [1, 2, 4]
n = 3
print(insert(list, n))
```

9. Write a program to create a queue and display all the members and size of the queue
   Solution:

```
import queue
q = queue.Queue()
for x in range(4):
    q.put(x)
print("Members of the queue:")
y=z=q.qsize()
for n in list(q.queue):
    print(n, end=" ")
print("\nSize of the queue:")
print(q.qsize())
```

10. Write a program to find whether a queue is empty or not.
    Solution:

```
import queue
p = queue.Queue()
q = queue.Queue()
for x in range(4):
    q.put(x)
print(p.empty())
print(q.empty())
```

11. Write a program to create a FIFO queue
    Solution:

```
import queue
q = queue.Queue()
for x in range(4):
```

```
    q.put(str(x))
while not q.empty():
    print(q.get(), end=" ")
print("\n")
```

12. Write a program to create a LIFO queue.
    Solution:

```
import queue
q = queue.LifoQueue()
for x in range(4):
    q.put(str(x))
while not q.empty():
    print(q.get(), end=" ")
print()
```

13. Write a program to find length of the longest substring of a given string without repeating characters.
    Solution:

```
class Solution(object):
  def lengthOfLongestSubstring(self, s):
    i =0
    j = 0
    d={}
    ans = 0
    while j < len(s):
      if s[j] not in d or i>d[s[j]]:
        ans = max(ans,(j-i+1))
        d[s[j]] = j
      else:
        i = d[s[j]]+1
        ans = max(ans,(j-i+1))
        j-=1
      #print(ans)
      j+=1
    return ans
ob1 = Solution()
print(ob1.lengthOfLongestSubstring("ABCABCBB"))
```

14. Write a program to find the median of the two given sorted arrays which are not empty.
    Solution:

```
def Solution(arr):
        n = len(arr)
        if n % 2 == 0:
                z = n // 2
                e = arr[z]
                q = arr[z - 1]
                ans = (e + q) / 2
                return ans
        else:
                z = n // 2
                ans = arr[z]
                return ans
if __name__ == "__main__":
        arr1 = [ -5, 3, 6, 12, 15 ]
        arr2 = [ -12, -10, -6, -3, 4, 10 ]
        arr3 = arr1 + arr2
        arr3.sort()
        print("Median = ", Solution(arr3))
```

15. Write a program to find the longest palindromic substring of a given string. Maximum length of the given string is 1000.
    Solution:

```
def longestPalSubstr(string):
        n = len(string)
        if (n < 2):
                return n
        start=0
        maxLength = 1
        for i in range(n):
                low = i - 1
                high = i + 1
                while (high < n and string[high] == string[i] ):
                        high=high+1
```

```
            while (low >= 0 and string[low] == string[i] ):
                low=low-1
        while (low >= 0 and high < n and string[low] == string[high] ):
            low=low-1
            high=high+1
            length = high - low - 1
            if (maxLength < length):
                maxLength = length
                start=low+1
        print ("Longest palindrome substring is:",end=" ")
        print (string[start:start + maxLength])
        return maxLength
        string = ("forgeeksskeegfor")
print("Length is: " + str(longestPalSubstr(string)))
```

16. Write a program to reverse digits of a given a 32-bit signed integer.
    Solution:

```
n = 4562
rev = 0
while(n > 0):
    a = n % 10
    rev = rev * 10 + a
    n = n // 10
print(rev)
```

17. Write a program to convert a given integer to roman number.
    Solution:

```
def printRoman(number):
    num = [1, 4, 5, 9, 10, 40, 50, 90,
        100, 400, 500, 900, 1000]
    sym = ["I", "IV", "V", "IX", "X", "XL",
        "L", "XC", "C", "CD", "D", "CM", "M"]
    i = 12
    while number:
        div = number // num[i]
        number %= num[i]
        while div:
```

```
                print(sym[i], end = "")
                div -= 1
            i -= 1
if __name__ == "__main__":
    number = 3549
    print("Roman value is:", end = " ")
    printRoman(number)
```

18. Write a programming to convert a given roman number to an integer.
    Solution:

```
def value(r):
    if (r == 'I'):
        return 1
    if (r == 'V'):
        return 5
    if (r == 'X'):
        return 10
    if (r == 'L'):
        return 50
    if (r == 'C'):
        return 100
    if (r == 'D'):
        return 500
    if (r == 'M'):
        return 1000
    return -1
def romanToDecimal(str):
    res = 0
    i = 0
    while (i < len(str)):
        s1 = value(str[i])
        if (i + 1 < len(str)):
            s2 = value(str[i + 1])
            if (s1 >= s2):
                res = res + s1
                i = i + 1
            else:
                res = res + s2 - s1
                i = i + 2
```

```
            else:
                res = res + s1
                i = i + 1
        return res
print("Integer form of Roman Numeral is"),
print(romanToDecimal("MCMIV"))
```

19. Write a program to find all unique triplets in a given array integers whose sum equal to zero.
    Solution:

```
def findTriplets(arr, n):
    found = False
    for i in range(0, n-2):
        for j in range(i+1, n-1):
            for k in range(j+1, n):
                if (arr[i] + arr[j] + arr[k] == 0):
                    print(arr[i], arr[j], arr[k])
                    found = True
    if (found == False):
        print(" not exist ")
arr = [0, -1, 2, -3, 1]
n = len(arr)
findTriplets(arr, n)
```

20. Write a program to find all unique quadruplets in a given array of integers whose sum equal to zero.
    Solution:

```
def countSum(a, n, sum):
    count = 0
    for i in range(n - 3):

        for j in range(i + 1, n - 2):

            for k in range(j + 1, n - 1):

                for l in range(k + 1, n):
                    if (a[i] + a[j] + a[k] + a[l]== sum):

                        count += 1
```

```
        return count
if __name__ == '__main__':

        arr = [ 4, 5, 3, 1, 2, 4 ]
        S = 13
        N = len(arr)
        print(countSum(arr, N, S))
```