Python Programming and Data Structure Programs (20 Marks programs)

1. Write a python program to perform following operations on BST.
   Insert
   Display

```
class Node:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.val = key

def insert(root, key):
    if root is None:
        return Node(key)
    if key < root.val:
        root.left = insert(root.left, key)
    else:
        root.right = insert(root.right, key)
    return root

def inorder(root):
    if root:
        inorder(root.left)
        print(root.val, end=" ")
        inorder(root.right)

# Example usage
root = None
root = insert(root, 50)
root = insert(root, 30)
root = insert(root, 70)
root = insert(root, 20)
root = insert(root, 40)
root = insert(root, 60)
root = insert(root, 80)
print("Inorder Traversal:")
inorder(root)
```

2. Write Python program to merge two sorted linked lists.

```
class ListNode:
    def __init__(self, value=0, next=None):
        self.value = value
        self.next = next

def merge_lists(l1, l2):
    if not l1 or not l2:
        return l1 or l2
    if l1.value < l2.value:
        l1.next = merge_lists(l1.next, l2)
        return l1
    else:
        l2.next = merge_lists(l1, l2.next)
        return l2

# Example usage
l1 = ListNode(1, ListNode(3, ListNode(5)))
l2 = ListNode(2, ListNode(4, ListNode(6)))
result = merge_lists(l1, l2)

while result:
    print(result.value, end=" -> ")
    result = result.next
print("None")
```

3. Write a python program to perform following operations on BST.
   Create
   Search
   Display (Preorder / Inorder / Postorder)

```
def preorder(root):
    if root:
        print(root.val, end=" ")
```

```
        preorder(root.left)
        preorder(root.right)

def postorder(root):
    if root:
        postorder(root.left)
        postorder(root.right)
        print(root.val, end=" ")

# Example usage
print("Preorder Traversal:")
preorder(root)
print("\nPostorder Traversal:")
postorder(root)
```

4. Python program for static implementation of Singly Linked List to perform Insert and Display operations.

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class SinglyLinkedList:
    def __init__(self):
        self.head = None

    def insert(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node

    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")

# Example usage
```

```
sll = SinglyLinkedList()
sll.insert(10)
sll.insert(20)
sll.insert(30)
sll.display()
```

5. Write a python program to perform following operations on Binary Search Tree
   i. Create
   ii. Count non-leaf nodes
   iii. Traversal (Prorder / Inorder / Postorder)

```
def count_non_leaf_nodes(root):
    if not root or (not root.left and not root.right):
        return 0
    return 1 + count_non_leaf_nodes(root.left) + count_non_leaf_nodes(root.right)

# Example usage
print("Non-Leaf Nodes Count:", count_non_leaf_nodes(root))
```

6. Python program for dynamic implementation of Singly Linked List to perform Insert and Display operations.

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class SinglyLinkedList:
    def __init__(self):
        self.head = None

    def insert_at_end(self, data):
        new_node = Node(data)
```

```
        if not self.head:
            self.head = new_node
            return
        current = self.head
        while current.next:
            current = current.next
        current.next = new_node

    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")

# Example usage
dll = SinglyLinkedList()
dll.insert_at_end(10)
dll.insert_at_end(20)
dll.insert_at_end(30)
dll.display()
```

7. Write a python program to perform following operations on Binary Search Tree

   Create

   i. Count leaf nodes

   ii. Traversal (Prorder / Inorder / Postorder)

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class SortedLinkedList:
```

```
    def __init__(self):
        self.head = None

    def insert_sorted(self, data):
        new_node = Node(data)
        if not self.head or self.head.data >= data:
            new_node.next = self.head
            self.head = new_node
        else:
            current = self.head
            while current.next and current.next.data < data:
                current = current.next
            new_node.next = current.next
            current.next = new_node

    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")

# Example usage
sll = SortedLinkedList()
sll.insert_sorted(30)
sll.insert_sorted(10)
sll.insert_sorted(20)
sll.display()
```

8. Python program to create a linked list in the sorted order.

9. Write a python program to perform following operations on BST
    i. Create
    ii.Delete
    iii. Traversal (Prorder / Inorder / Postorder)

```
  def delete_node(root, key):
    if not root:
        return root
    if key < root.val:
        root.left = delete_node(root.left, key)
    elif key > root.val:
        root.right = delete_node(root.right, key)
    else:
        if not root.left:
            return root.right
        elif not root.right:
            return root.left
        temp = find_min(root.right)
        root.val = temp.val
        root.right = delete_node(root.right, temp.val)
    return root

def find_min(node):
    while node.left:
        node = node.left
    return node

# Example usage (continuing from previous BST code)
root = delete_node(root, 30)
print("Inorder Traversal After Deletion:")
inorder(root)
```

10. Write a python program for implementation of Doubly Linked List to perform Insert and Display operations.

```
    class DNode:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None

class DoublyLinkedList:
    def __init__(self):
```

```
        self.head = None

    def insert(self, data):
        new_node = DNode(data)
        if not self.head:
            self.head = new_node
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node
            new_node.prev = current

    def display(self):
        current = self.head
        while current:
            print(current.data, end=" <-> ")
            current = current.next
        print("None")

# Example usage
dll = DoublyLinkedList()
dll.insert(10)
dll.insert(20)
dll.insert(30)
dll.display()
```

11. Write a python program to perform following operations on Binary Search Tree
    i. Create
    ii. Count total nodes
    iii. Traversal (Prorder / Inorder / Postorder)

```
def count_total_nodes(root):
    if not root:
        return 0
    return 1 + count_total_nodes(root.left) + count_total_nodes(root.right)

# Example usage
print("Total Nodes Count:", count_total_nodes(root))
```

12. Python program to create doubly linked list and search the given node in the Linked list.

```
def search_dll(head, key):
    current = head
    while current:
        if current.data == key:
            return True
        current = current.next
    return False

# Example usage
print("Search for 20 in Doubly Linked List:", search_dll(dll.head, 20))
```

13. Write a python program to perform following operations on BST.
    i. Create
    ii. Display

14. Python program to create singly linked list and search the given node in the Linked list.

```
    def search_sll(head, key):
    current = head
    while current:
        if current.data == key:
            return True
        current = current.next
    return False

# Example usage
print("Search for 20 in Singly Linked List:", search_sll(sll.head, 20))
```

15. Write a python program to perform following operations on BST.
    i. Insert
    ii.Delete
    iii.Display (Preorder / Inorder / Postorder)
    ans on 1 and 9
16. Python program to create singly linked list and reverse the Linked list.

```
def reverse_sll(head):
    prev = None
```

```
    current = head
    while current:
        next_node = current.next
        current.next = prev
        prev = current
        current = next_node
    return prev

# Example usage
sll.head = reverse_sll(sll.head)
print("Reversed Linked List:")
sll.display()
```

17. Write a program to search an element using Linear Search.

```
def linear_search(arr, key):
    for i, val in enumerate(arr):
        if val == key:
            return i
    return -1

# Example usage
arr = [10, 20, 30, 40]
key = 30
print(f"Element {key} found at index:", linear_search(arr, key))
```

18. Write a program to calculate indegree of a graph using adjacency matrix.

```
def calculate_indegree(graph):
    return [sum(row) for row in zip(*graph)]

# Example usage
graph = [
    [0, 1, 0],
    [0, 0, 1],
    [1, 0, 0]
]
print("Indegree of Graph:", calculate_indegree(graph))
```

19. Write a program to search an element using Binary Search.

```
def binary_search(arr, key):
    low, high = 0, len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == key:
            return mid
        elif arr[mid] < key:
            low = mid + 1
        else:
            high = mid - 1
    return -1

# Example usage
arr = [10, 20, 30, 40, 50]
key = 30
print(f"Element {key} found at index:", binary_search(arr, key))
```

20. Write a Python program to calculate outdegree of a graph using adjacency matrix.

```
def calculate_outdegree(graph):
    return [sum(row) for row in graph]

# Example graph as an adjacency matrix
# Graph:
#   0 --> 1
#   1 --> 2
#   2 --> 0
graph = [
    [0, 1, 0],  # Node 0 has an edge to Node 1
    [0, 0, 1],  # Node 1 has an edge to Node 2
    [1, 0, 0]   # Node 2 has an edge to Node 0
]

print("Outdegree of each node:", calculate_outdegree(graph))
```

21. Write a Python program to sort given numbers using Bubble Sort algorithms.

```
def bubble_sort(arr):
```

```
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# Example usage
arr = [64, 34, 25, 12, 22, 11, 90]
print("Sorted Array:", bubble_sort(arr))
```

22. Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle

```
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14159 * self.radius * self.radius

    def perimeter(self):
        return 2 * 3.14159 * self.radius

# Example usage
circle = Circle(5)
print("Area:", circle.area())
print("Perimeter:", circle.perimeter())
```

23. Write a Python program to implement sorting Merge Sort algorithms.

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left = arr[:mid]
        right = arr[mid:]

        merge_sort(left)
        merge_sort(right)

        i = j = k = 0
```

```
        while i < len(left) and j < len(right):
            if left[i] < right[j]:
                arr[k] = left[i]
                i += 1
            else:
                arr[k] = right[j]
                j += 1
            k += 1

        while i < len(left):
            arr[k] = left[i]
            i += 1
            k += 1

        while j < len(right):
            arr[k] = right[j]
            j += 1
            k += 1

# Example usage
arr = [12, 11, 13, 5, 6, 7]
merge_sort(arr)
print("Sorted Array:", arr)
```

24. Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.

```
class ShoppingCart:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price, quantity):
        if name in self.items:
            self.items[name]['quantity'] += quantity
        else:
            self.items[name] = {'price': price, 'quantity': quantity}

    def remove_item(self, name):
        if name in self.items:
            del self.items[name]

    def total_price(self):
        return sum(item['price'] * item['quantity'] for item in self.items.values())
```

```
# Example usage
cart = ShoppingCart()
cart.add_item("Apple", 10, 3)
cart.add_item("Banana", 5, 2)
print("Total Price:", cart.total_price())
```

25. Write a Python program to implement sorting Quick Sort algorithms.

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

# Example usage
arr = [10, 7, 8, 9, 1, 5]
print("Sorted Array:", quick_sort(arr))
```

26. Write a Python program to implement sorting Insertion Sort algorithms.

```
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

# Example usage
arr = [12, 11, 13, 5, 6]
print("Sorted Array:", insertion_sort(arr))
```

27. Write a Python program to calculate indegree of a graph.

28. Write a Python program to calculate outdegree of a graph.

29. Write a python code for static implementation of stack.

```
class Stack:
    def __init__(self, size):
        self.stack = []
        self.size = size

    def push(self, item):
        if len(self.stack) < self.size:
            self.stack.append(item)
        else:
            print("Stack Overflow")

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()
        print("Stack Underflow")
        return None

    def is_empty(self):
        return len(self.stack) == 0

# Example usage
stack = Stack(5)
stack.push(10)
stack.push(20)
print("Popped:", stack.pop())
```

30. Write a Python program for Evaluation of postfix expression.

```
def evaluate_postfix(expression):
    stack = []
    for char in expression:
        if char.isdigit():
            stack.append(int(char))
        else:
            b = stack.pop()
            a = stack.pop()
            if char == '+':
```

```
            stack.append(a + b)
        elif char == '-':
            stack.append(a - b)
        elif char == '*':
            stack.append(a * b)
        elif char == '/':
            stack.append(a // b)
    return stack.pop()

# Example usage
expression = "231*+9-"
print("Result:", evaluate_postfix(expression))
```

31. Write a python code for static implementation of queue.

```
class Queue:
    def __init__(self, size):
        self.queue = []
        self.size = size

    def enqueue(self, item):
        if len(self.queue) < self.size:
            self.queue.append(item)
        else:
            print("Queue Overflow")

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)
        print("Queue Underflow")
        return None

    def is_empty(self):
        return len(self.queue) == 0

# Example usage
queue = Queue(5)
queue.enqueue(10)
queue.enqueue(20)
print("Dequeued:", queue.dequeue())
```

32. Write a python code for dynamic implementation of Stack to perform following operations: Init,

Push, Pop, Isempty, Isfull.

```
class DynamicStack:
    def __init__(self):
        self.stack = []

    def push(self, item):
        self.stack.append(item)

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()
        print("Stack Underflow")
        return None

    def is_empty(self):
        return len(self.stack) == 0

    def is_full(self):
        # For dynamic stack, it cannot be full
        return False

# Example usage
stack = DynamicStack()
stack.push(10)
stack.push(20)
print("Popped:", stack.pop())
```

33. Write a python code for simple implementation of priority queue.

```
import heapq

class PriorityQueue:
    def __init__(self):
        self.queue = []

    def push(self, item):
        heapq.heappush(self.queue, item)

    def pop(self):
        return heapq.heappop(self.queue) if not self.is_empty() else None

    def is_empty(self):
        return len(self.queue) == 0
```

```
# Example usage
pq = PriorityQueue()
pq.push(3)
pq.push(1)
pq.push(2)
print("Popped:", pq.pop())
```

34. Write a Python program to convert infix to postfix conversion using stack.

```
def precedence(op):
    if op in ('+', '-'):
        return 1
    if op in ('*', '/'):
        return 2
    return 0

def infix_to_postfix(expression):
    stack = []
    result = []
    for char in expression:
        if char.isalnum():
            result.append(char)
        elif char == '(':
            stack.append(char)
        elif char == ')':
            while stack and stack[-1] != '(':
                result.append(stack.pop())
            stack.pop()
        else:
            while stack and precedence(char) <= precedence(stack[-1]):
                result.append(stack.pop())
            stack.append(char)
    while stack:
        result.append(stack.pop())
    return ''.join(result)

# Example usage
expression = "a+b*(c^d-e)^(f+g*h)-i"
print("Postfix Expression:", infix_to_postfix(expression))
```

35. Write a python code for simple implementation of priority queue.

36. Write a python code for dynamic implementation of linear Queue to perform following operations: init, enqueue, dequeue, isEmpty, isFull.

```
class DynamicQueue:
    def __init__(self):
        self.queue = []

    def enqueue(self, item):
        self.queue.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)
        print("Queue Underflow")
        return None

    def is_empty(self):
        return len(self.queue) == 0

# Example usage
dq = DynamicQueue()
dq.enqueue(10)
dq.enqueue(20)
print("Dequeued:", dq.dequeue())
```

37. Write a python code for Implementation of an algorithm that reverses string of characters using stack and checks whether a string is a palindrome or not.

```
def is_palindrome(s):
    stack = list(s)
    reversed_s = ''.join(stack[::-1])
    return s == reversed_s

# Example usage
string = "radar"
print(f"Is '{string}' a palindrome?:", is_palindrome(string))
```

38. Write a python code for implementation of circular queue.

```
class CircularQueue:
    def __init__(self, size):
        self.size = size
        self.queue = [None] * size
        self.front = self.rear = -1

    def enqueue(self, item):
        if (self.rear + 1) % self.size == self.front:
            print("Queue Overflow")
            return
        if self.front == -1:
            self.front = self.rear = 0
        else:
            self.rear = (self.rear + 1) % self.size
        self.queue[self.rear] = item

    def dequeue(self):
        if self.front == -1:
            print("Queue Underflow")
            return None
        temp = self.queue[self.front]
        if self.front == self.rear:
            self.front = self.rear = -1
        else:
            self.front = (self.front + 1) % self.size
        return temp

    def display(self):
        if self.front == -1:
            print("Queue is Empty")
            return
        idx = self.front
        while True:
            print(self.queue[idx], end=" ")
            if idx == self.rear:
                break
            idx = (idx + 1) % self.size
        print()

# Example usage
cq = CircularQueue(5)
cq.enqueue(10)
cq.enqueue(20)
cq.dequeue()
cq.display()
```