Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

1

Q.1 Write a Python program to search an element within an array using linear search.

[10 Mks]

⇨ def linear_search(arr, target):

```
   for i in range(len(arr)):
      if arr[i] == target:
         return i  # Return the index of the target if found
   return -1  # Return -1 if the target is not found

# Example usage:
arr = [10, 23, 45, 70, 8, 90]
target = 70

result = linear_search(arr, target)
if result != -1:
   print(f"Element {target} found at index {result}")
else:
   print(f"Element {target} not found in the array.")
```

Q.2 Write a Python program to perform infix to postfix conversion of given expression using stack.

[20 Mks]

⇨ # Function to check if the character is an operator

```
def is_operator(c):
   return c in '+-*/^'
```

```
# Function to get precedence of operators
def precedence(c):
    if c in '+-':
        return 1
    if c in '*/':
        return 2
    if c == '^':
        return 3
    return 0

# Function to convert infix to postfix using stack
def infix_to_postfix(infix):
    stack = []
    postfix = []

    for char in infix:
        if char.isalnum():  # If character is operand (number or letter)
            postfix.append(char)
        elif char == '(':  # Left parenthesis
            stack.append(char)
        elif char == ')':  # Right parenthesis
            while stack and stack[-1] != '(':
                postfix.append(stack.pop())
            stack.pop()  # Remove '(' from stack
        elif is_operator(char):  # If character is an operator
            while stack and precedence(stack[-1]) >= precedence(char):
                postfix.append(stack.pop())
            stack.append(char)

    while stack:
        postfix.append(stack.pop())
```

```
    return ''.join(postfix)

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

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

2

Q. 1 Write a Python program to search an element within an array using binary search.

[10 Mks]

⇨ def binary_search(arr, target):

```
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid  # Return the index of the target
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1  # Return -1 if target is not found

# Example usage:
```

```python
arr = [1, 3, 5, 7, 9, 11, 13]
target = 7

result = binary_search(arr, target)
if result != -1:
    print(f"Element {target} found at index {result}")
else:
    print(f"Element {target} not found in the array.")
```

Q. 2 Write a Python program to evaluate postfix expression using stack.

[20 Mks]

⇨ def evaluate_postfix(expression):

```python
    stack = []
    for char in expression:
        if char.isdigit():  # If character is a number
            stack.append(int(char))
        else:  # If character is an operator
            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)
            elif char == '^':
                stack.append(a ** b)
    return stack[-1]  # The result will be the last element in the stack
```

Example usage:

expression = "23*5+"

print(f"Result of postfix expression {expression}: {evaluate_postfix(expression)}")

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

3

Q. 1 Write a Python program to sort array elements using bubble sort algorithm.

[10 Mks]

⇨ 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]  # Swap if elements are in wrong order
    return arr

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

Q.2 Write a Python program for dynamic implementation of Singly Linked List to perform following operations:

a. Create

b. Display

c. Search

[20 Mks]

⇨ # Node class to represent each element in the linked list

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

# Singly Linked List class
class SinglyLinkedList:
    def __init__(self):
        self.head = None

    # Function to create a new node and insert it at the end
    def insert(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
        last = self.head
        while last.next:
            last = last.next
        last.next = new_node

    # Function to display the elements of the linked list
    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
```

```
        print("None")

    # Function to search for an element in the linked list
    def search(self, key):
        current = self.head
        while current:
            if current.data == key:
                return True
            current = current.next
        return False

# Example usage:
linked_list = SinglyLinkedList()
linked_list.insert(10)
linked_list.insert(20)
linked_list.insert(30)
linked_list.display()

# Searching for a value
if linked_list.search(20):
    print("Element found.")
else:
    print("Element not found.")
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

4

Q. 1 Write a Python program to sort array elements using insertion sort algorithm.

[10 Mks]

⇨ def insertion_sort(arr):

```
    for i in range(1, len(arr)):  # Iterate over the array starting from index 1
        key = arr[i]  # The element to be placed in the sorted portion
        j = i - 1
        # Shift elements of the sorted portion to the right to make space for key
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key  # Insert the key at the correct position
    return arr

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

Q.2 Write a Python program for dynamic implementation of Doubly Circular Linked List to perform following operations:

a. Create

b. Display

[20 Mks]

⇨ # Node class for the doubly circular linked list

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

```
# Doubly Circular Linked List class
class DoublyCircularLinkedList:
    def __init__(self):
        self.head = None

    # Function to create and insert a new node at the end of the list
    def insert(self, data):
        new_node = Node(data)
        if not self.head:  # If the list is empty
            self.head = new_node
            new_node.next = new_node  # Point to itself to form a circular link
            new_node.prev = new_node
        else:
            last = self.head.prev  # Get the last node
            last.next = new_node
            new_node.prev = last
            new_node.next = self.head
            self.head.prev = new_node

    # Function to display the elements of the list
    def display(self):
        if not self.head:
            print("List is empty")
            return
        current = self.head
        while True:
            print(current.data, end=" <=> ")
            current = current.next
            if current == self.head:
                break
```

```
    print("Circular list end")

# Example usage:
cdll = DoublyCircularLinkedList()
cdll.insert(10)
cdll.insert(20)
cdll.insert(30)
cdll.display()  # Output: 10 <=> 20 <=> 30 <=> Circular list end
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

5

Q.1 Write a Python program to sort array elements using merge sort algorithm.

[10 Mks]

⇨
```
def merge_sort(arr):
  if len(arr) > 1:
    mid = len(arr) // 2  # Find the middle point
    left_half = arr[:mid]
    right_half = arr[mid:]

    merge_sort(left_half)  # Sort the left half
    merge_sort(right_half)  # Sort the right half

    i = j = k = 0
```

```
        # Merge the sorted halves
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        # Copy the remaining elements from the left half (if any)
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        # Copy the remaining elements from the right half (if any)
        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

    return arr

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

Q.2 Write a Python program for static implementation of linear queue to perform following operations:

a. init

b. enqueue

c. dequeue

d. isEmpty

e. isFull

[20 Mks]

⇨ class Queue:

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

    def is_empty(self):
        return self.front == -1

    def is_full(self):
        return self.rear == self.size - 1

    def enqueue(self, value):
        if self.is_full():
            print("Queue is full, cannot enqueue!")
            return
        if self.front == -1:
            self.front = 0
        self.rear += 1
        self.queue[self.rear] = value
        print(f"Enqueued: {value}")

    def dequeue(self):
        if self.is_empty():
```

```
            print("Queue is empty, cannot dequeue!")
            return None
        dequeued_value = self.queue[self.front]
        self.front += 1
        if self.front > self.rear:  # Reset the queue after dequeue
            self.front = self.rear = -1
        print(f"Dequeued: {dequeued_value}")
        return dequeued_value

# Example usage:
queue = Queue(5)
queue.enqueue(10)
queue.enqueue(20)
queue.dequeue()
queue.enqueue(30)
queue.enqueue(40)
queue.enqueue(50)
queue.enqueue(60)  # Queue is full, should print error
```

Q. 3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

6

Q.1 Write a Python program to sort array elements using quick sort algorithm.

[10 Mks]

⇨ def quick_sort(arr):

```
    if len(arr) <= 1:
```

```
        return arr
    pivot = arr[len(arr) // 2]  # Choose the middle element as pivot
    left = [x for x in arr if x < pivot]  # Elements less than pivot
    middle = [x for x in arr if x == pivot]  # Elements equal to pivot
    right = [x for x in arr if x > pivot]  # Elements greater than pivot
    return quick_sort(left) + middle + quick_sort(right)

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

Q.2 Write a Python program for dynamic implementation of Singly Linked List to perform following operations:

a. Create

b. Display

c. Merge

[20 Mks]

⇨ # Node class for Singly Linked List

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

# Singly Linked List class
class SinglyLinkedList:
    def __init__(self):
        self.head = None

    # Method to create and add a new node
    def create(self, data):
```

```
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node

    # Method to display the list
    def display(self):
        current = self.head
        if not current:
            print("List is empty")
            return
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")

    # Method to merge two sorted linked lists
    def merge(self, other_list):
        merged = SinglyLinkedList()
        current1 = self.head
        current2 = other_list.head

        while current1 and current2:
            if current1.data <= current2.data:
                merged.create(current1.data)
                current1 = current1.next
            else:
                merged.create(current2.data)
                current2 = current2.next

        # Add remaining elements if any
        while current1:
```

```
            merged.create(current1.data)
            current1 = current1.next
        while current2:
            merged.create(current2.data)
            current2 = current2.next

        return merged

# Example usage:
list1 = SinglyLinkedList()
list1.create(10)
list1.create(5)
list1.create(3)
list1.display()

list2 = SinglyLinkedList()
list2.create(20)
list2.create(15)
list2.display()

merged_list = list1.merge(list2)
merged_list.display()  # Merged list should be sorted
```

Q. 3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

7

Q.1 Write a Python program to check whether a given string is palindrome or not using stack.

[10 Mks]

⇨ # Function to check if a string is a palindrome using a stack

```
def is_palindrome(s):
   stack = []
   # Push each character of the string onto the stack
   for char in s:
      stack.append(char)

   # Compare each character with the original string
   for char in s:
      if char != stack.pop():  # Pop from stack and check
         return False
   return True

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

Q.2 Write a Python program for dynamic implementation of linear queue to perform following operations:

a. init

b. enqueue

c. dequeue

d. isEmpty

[20 Mks]

⇨ # Queue class for dynamic linear queue implementation

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

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

    def is_full(self):
        return len(self.queue) == self.size

    def enqueue(self, value):
        if self.is_full():
            print("Queue is full, cannot enqueue!")
        else:
            self.queue.append(value)
            print(f"Enqueued: {value}")

    def dequeue(self):
        if self.is_empty():
            print("Queue is empty, cannot dequeue!")
            return None
        dequeued_value = self.queue.pop(0)  # Remove the front element
        print(f"Dequeued: {dequeued_value}")
        return dequeued_value

# Example usage:
queue = Queue(5)
queue.enqueue(10)
queue.enqueue(20)
queue.dequeue()
queue.enqueue(30)
queue.enqueue(40)
queue.enqueue(50)
queue.enqueue(60)  # Queue is full, should print error
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

8

Q.1 Write a python function to calculate the factorial of a number. The function should accept the number as an argument. [10 Mks]

⇨ def factorial(n):

```
  if n == 0 or n == 1:
    return 1
  return n * factorial(n - 1)

# Example usage:
num = 5
print(f"Factorial of {num}: {factorial(num)}")
```

Q.2 Write a Python program for static implementation of stack to perform following operations:

a. init

b. push

c. pop

d. isEmpty

e. isFull

[20 Mks]

⇨ class Stack:

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

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

    def is_full(self):
        return len(self.stack) == self.size

    def push(self, value):
        if self.is_full():
            print("Stack is full, cannot push!")
        else:
            self.stack.append(value)
            print(f"Pushed: {value}")

    def pop(self):
        if self.is_empty():
            print("Stack is empty, cannot pop!")
            return None
        return self.stack.pop()

# Example usage:
stack = Stack(3)
stack.push(10)
stack.push(20)
stack.push(30)
stack.push(40)  # Stack is full, should print error
print(f"Popped: {stack.pop()}")
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

9

Q.1 Write a program which accepts 6 integer values and prints “DUPLICATES” if any of the values entered are duplicates otherwise it prints “ALL UNIQUE”. Example: Let 5 integers are (32, 10, 45, 90, 45, 6) then output “DUPLICATES” to be printed. [10 Mks]

⇨ def check_duplicates(nums):

```
  if len(nums) != len(set(nums)):  # If list length differs from set length, there are duplicates
    print("DUPLICATES")
  else:
    print("ALL UNIQUE")

# Example usage:
nums = [32, 10, 45, 90, 45, 6]
check_duplicates(nums)  # Output: DUPLICATES
```

Q.2 Write a Python program for dynamic implementation of stack to perform following operations:

a. init

b. push

c. pop

d. isEmpty

[20 Mks]

⇨ class Stack:

```
  def __init__(self):
    """Initialize an empty stack."""
    self.stack = []

  def push(self, value):
```

```
        """Push an element onto the stack."""
        self.stack.append(value)
        print(f"Pushed: {value}")

    def pop(self):
        """Pop the top element from the stack."""
        if self.is_empty():
            print("Stack is empty, cannot pop!")
            return None
        popped_value = self.stack.pop()
        print(f"Popped: {popped_value}")
        return popped_value

    def is_empty(self):
        """Check if the stack is empty."""
        return len(self.stack) == 0

# Example usage:
stack = Stack()

# Pushing elements onto the stack
stack.push(10)
stack.push(20)
stack.push(30)

# Popping elements from the stack
stack.pop()

# Checking if the stack is empty
if stack.is_empty():
    print("The stack is empty.")
```

else:

    print("The stack is not empty.")

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

10

Dictionary:

Q.1 Write a Python program to combine two dictionary adding values for common keys. Sample

d1={'a':100,'b':200,'c':300}

d2={'a':300,'b':200,'d':400}

Sample output: Counter ({'a': 400, 'b': 400, 'd': 400, 'c': 300})

⇨ from collections import Counter

d1 = {'a': 100, 'b': 200, 'c': 300}

d2 = {'a': 300, 'b': 200, 'd': 400}

Combine dictionaries, adding values for common keys

result = dict(Counter(d1) + Counter(d2))

print(result) # Output: {'a': 400, 'b': 400, 'c': 300, 'd': 400}

Q.2 Write a Python program for dynamic implementation of Singly Linked List to perform following operations:

a. Create

b. Display

c. Sort

[20 Mks]

⇨ class Node:

```
    def __init__(self, data):
        """Initialize a node with data and next pointer."""
        self.data = data
        self.next = None

class SinglyLinkedList:
    def __init__(self):
        """Initialize the head of the linked list."""
        self.head = None

    def create(self, data):
        """Create a node and insert it at the end of the list."""
        new_node = Node(data)
        if not self.head:
            self.head = new_node
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node
        print(f"Inserted: {data}")

    def display(self):
        """Display all elements in the list."""
        if not self.head:
            print("The list is empty.")
            return
        current = self.head
        print("Linked List: ", end="")
        while current:
            print(current.data, end=" -> ")
```

```
            current = current.next
        print("None")

    def sort(self):
        """Sort the elements in the linked list in ascending order."""
        if not self.head or not self.head.next:
            print("No need to sort, the list has 0 or 1 element.")
            return

        # Convert linked list to a Python list to sort
        current = self.head
        data_list = []
        while current:
            data_list.append(current.data)
            current = current.next

        # Sort the data list
        data_list.sort()

        # Rebuild the linked list with sorted data
        current = self.head
        for data in data_list:
            current.data = data
            current = current.next

        print("List sorted in ascending order.")

# Example usage:
linked_list = SinglyLinkedList()

# Create linked list
```

```
linked_list.create(10)
linked_list.create(5)
linked_list.create(30)
linked_list.create(15)

# Display the linked list
linked_list.display()

# Sort the linked list
linked_list.sort()

# Display the sorted linked list
linked_list.display()
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

11

Q.1 Write a Python program for static implementation of Binary Tree.

[10 Mks]

⇨ class Node:

```
  def __init__(self, data):
    self.data = data
    self.left = None
    self.right = None
```

```
class BinaryTree:
    def __init__(self, root_data):
        self.root = Node(root_data)

    def add_left(self, parent_data, data):
        parent_node = self.find_node(self.root, parent_data)
        if parent_node:
            parent_node.left = Node(data)
        else:
            print("Parent node not found.")

    def add_right(self, parent_data, data):
        parent_node = self.find_node(self.root, parent_data)
        if parent_node:
            parent_node.right = Node(data)
        else:
            print("Parent node not found.")

    def find_node(self, node, data):
        if node is None:
            return None
        if node.data == data:
            return node
        left_result = self.find_node(node.left, data)
        if left_result:
            return left_result
        return self.find_node(node.right, data)

    def display(self):
        self._inorder_traversal(self.root)
```

```python
    def _inorder_traversal(self, node):
        if node:
            self._inorder_traversal(node.left)
            print(node.data, end=" ")
            self._inorder_traversal(node.right)

# Example usage:
bt = BinaryTree(1)
bt.add_left(1, 2)
bt.add_right(1, 3)
bt.add_left(2, 4)
bt.add_right(2, 5)

bt.display()
```

Q.2 Write a Python program that accepts the vertices and edges of an undirected graph and stores it as an adjacency matrix. Display the adjacency matrix.

[20 Mks]

- ⇨ def create_adjacency_matrix(vertices, edges):

```python
    # Initialize an adjacency matrix with 0s
    matrix = [[0 for _ in range(vertices)] for _ in range(vertices)]

    for edge in edges:
        u, v = edge
        matrix[u][v] = 1
        matrix[v][u] = 1  # Since it's an undirected graph

    return matrix

def display_adjacency_matrix(matrix):
    for row in matrix:
```

```
        print(" ".join(map(str, row)))

# Example usage:
vertices = 5  # Number of vertices (0 to 4)
edges = [(0, 1), (0, 2), (1, 3), (2, 3), (3, 4)]  # Edges in the graph

adj_matrix = create_adjacency_matrix(vertices, edges)
print("Adjacency Matrix:")
display_adjacency_matrix(adj_matrix)
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

12

Q.1 Write a Python program for dynamic implementation of Binary Tree.

[10 Mks]

⇨ class Node:

```
    def __init__(self, key):
        self.left = None
        self.right = None
        self.data = key

class BinarySearchTree:
    def __init__(self):
        self.root = None
```

```python
    def insert(self, root, key):
        """Insert a node in the BST."""
        if root is None:
            return Node(key)
        else:
            if key < root.data:
                root.left = self.insert(root.left, key)
            else:
                root.right = self.insert(root.right, key)
        return root

    def inorder(self, root):
        """In-order traversal of the BST."""
        if root:
            self.inorder(root.left)
            print(root.data, end=" ")
            self.inorder(root.right)

# Example usage:
bst = BinarySearchTree()
root = None

# Insert elements into the BST
root = bst.insert(root, 50)
root = bst.insert(root, 30)
root = bst.insert(root, 20)
root = bst.insert(root, 40)
root = bst.insert(root, 70)
root = bst.insert(root, 60)
root = bst.insert(root, 80)
```

```
print("In-order traversal of the BST:")
bst.inorder(root)
```

Q.2 Write a Python program that accepts the vertices and edges of an undirected graph and store it as an adjacency matrix. Implement function to print degree of all vertices of graph.

[20 Mks]

⇨ def create_adjacency_matrix(vertices, edges):

```
  # Initialize an adjacency matrix with all zeros
  matrix = [[0 for _ in range(vertices)] for _ in range(vertices)]

  # Populate the adjacency matrix for the given edges
  for u, v in edges:
    matrix[u][v] = 1
    matrix[v][u] = 1  # Since it's an undirected graph

  return matrix

def print_adjacency_matrix(matrix):
  for row in matrix:
    print(" ".join(map(str, row)))

def print_degree_of_vertices(matrix):
  # Degree of a vertex is the sum of the values in its row (or column)
  degrees = [sum(row) for row in matrix]
  for i, degree in enumerate(degrees):
    print(f"Degree of vertex {i}: {degree}")

# Example usage:
vertices = 5  # Number of vertices (0 to 4)
edges = [(0, 1), (0, 2), (1, 3), (2, 3), (3, 4)]  # Edges of the graph
```

Create adjacency matrix

adj_matrix = create_adjacency_matrix(vertices, edges)

Display adjacency matrix

print("Adjacency Matrix:")

print_adjacency_matrix(adj_matrix)

Print degree of each vertex

print("\nDegree of vertices:")

print_degree_of_vertices(adj_matrix)

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

13

Q.1 Write a Python program for dynamic implementation of Binary Search Tree.

[10 Mks]

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

class BinarySearchTree:
    def __init__(self):
        self.root = None
```

```
    def insert(self, root, key):
        """Insert a node into the BST."""
        if root is None:
            return Node(key)
        else:
            if key < root.data:
                root.left = self.insert(root.left, key)
            else:
                root.right = self.insert(root.right, key)
        return root

    def inorder(self, root):
        """In-order traversal of the BST."""
        if root:
            self.inorder(root.left)
            print(root.data, end=" ")
            self.inorder(root.right)

    def search(self, root, key):
        """Search for a key in the BST."""
        if root is None or root.data == key:
            return root
        elif key < root.data:
            return self.search(root.left, key)
        else:
            return self.search(root.right, key)

# Example usage:
bst = BinarySearchTree()
root = None
```

```
# Insert elements into the BST
values = [50, 30, 20, 40, 70, 60, 80]
for value in values:
    root = bst.insert(root, value)

print("In-order traversal of the BST:")
bst.inorder(root)
print()  # For newline

# Search for a value in the BST
key = 40
found_node = bst.search(root, key)
if found_node:
    print(f"Node with value {key} found.")
else:
    print(f"Node with value {key} not found.")
```

Q.2 Write a Python program that accepts the vertices and edges of a directed graph. Create and display adjacency list.

[20 Mks]

⇨ def create_adjacency_list(vertices, edges):

```
    # Initialize the adjacency list for each vertex
    adj_list = {i: [] for i in range(vertices)}

    # Add directed edges to the adjacency list
    for u, v in edges:
        adj_list[u].append(v)

    return adj_list

def display_adjacency_list(adj_list):
```

```
    print("Adjacency List Representation of the Graph:")
    for vertex, neighbors in adj_list.items():
        print(f"Vertex {vertex} -> {', '.join(map(str, neighbors)) if neighbors else 'No neighbors'}")

# Example usage:
# Number of vertices in the directed graph
vertices = 5  # For example, vertices numbered 0, 1, 2, 3, 4

# List of edges (directed), each edge is represented as a tuple (u, v) where u -> v
edges = [(0, 1), (0, 2), (1, 3), (2, 3), (3, 4)]

# Create the adjacency list
adj_list = create_adjacency_list(vertices, edges)

# Display the adjacency list
display_adjacency_list(adj_list)
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

14

Q.1 Write a Python program to count Leaf nodes of Binary Search Tree.

[10 Mks]

⇨ # Definition of a Node in the Binary Search Tree

```
class Node:
    def __init__(self, key):
```

```
        self.left = None
        self.right = None
        self.value = key

# Function to insert a node into the BST
def insert(root, key):
    # If the tree is empty, return a new node
    if root is None:
        return Node(key)

    # Otherwise, recur down the tree
    if key < root.value:
        root.left = insert(root.left, key)
    else:
        root.right = insert(root.right, key)

    # return the (unchanged) node pointer
    return root

# Function to count leaf nodes in the BST
def count_leaf_nodes(root):
    # If the tree is empty, return 0
    if root is None:
        return 0

    # If a node is a leaf node (both left and right are None), return 1
    if root.left is None and root.right is None:
        return 1

    # Otherwise, count leaf nodes in both subtrees
    return count_leaf_nodes(root.left) + count_leaf_nodes(root.right)
```

```python
# Example usage:
if __name__ == "__main__":
    # Create a BST and insert nodes
    root = Node(50)
    root = insert(root, 30)
    root = insert(root, 20)
    root = insert(root, 40)
    root = insert(root, 70)
    root = insert(root, 60)
    root = insert(root, 80)

    # Count the leaf nodes
    leaf_count = count_leaf_nodes(root)
    print(f"The number of leaf nodes in the BST is: {leaf_count}")
```

Q.2 Write a Python program that accepts the vertices and edges of a directed graph. Create and display adjacency list. Implement functions to print in-degree of all vertices of graph.

[20 Mks]

⇨ # Class to represent a directed graph using adjacency list

```python
class Graph:
    def __init__(self, vertices):
        self.V = vertices  # Number of vertices
        self.adj_list = {i: [] for i in range(vertices)}  # Adjacency list for each vertex

    # Function to add an edge to the graph
    def add_edge(self, u, v):
        self.adj_list[u].append(v)

    # Function to display the adjacency list
    def display_adj_list(self):
```

```
        print("Adjacency List:")
        for vertex in self.adj_list:
            print(f"Vertex {vertex}: {self.adj_list[vertex]}")

    # Function to calculate and print the in-degree of each vertex
    def in_degree(self):
        in_degree_count = [0] * self.V  # Initialize in-degree for each vertex to 0

        # Traverse the adjacency list and calculate in-degree
        for u in range(self.V):
            for v in self.adj_list[u]:
                in_degree_count[v] += 1

        # Print the in-degree of each vertex
        print("\nIn-degree of each vertex:")
        for i in range(self.V):
            print(f"Vertex {i}: {in_degree_count[i]}")

# Example usage
if __name__ == "__main__":
    # Accept number of vertices and edges
    vertices = int(input("Enter number of vertices: "))
    edges = int(input("Enter number of edges: "))

    # Create a graph with the specified number of vertices
    graph = Graph(vertices)

    # Accept the edges
    print("Enter the edges (u, v) where there is an edge from vertex u to vertex v:")
    for _ in range(edges):
        u, v = map(int, input().split())
```

```
        graph.add_edge(u, v)

    # Display the adjacency list
    graph.display_adj_list()

    # Display the in-degree of each vertex
    graph.in_degree()
```

Q.3 Viva [5 Mks]

Savitribai Phule Pune University

M.Sc. (Computer Application) Sem-I University Practical Examination, December-2023

SUBJECT: CA 505 MJP-Lab Course Based on CA 502 MJ

(Python Programming and Data Structures)

Duration: 3 Hours Max. Marks: 35

15

Q.1 Write a Python program to count Non-Leaf nodes of Binary Search Tree.

[10 Mks]

⇨ # Node class to represent each node of the BST

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

# Binary Search Tree class
class BinarySearchTree:
    def __init__(self):
        self.root = None
```

```
# Function to insert a new node with the given key
def insert(self, key):
    if self.root is None:
        self.root = Node(key)
    else:
        self._insert(self.root, key)

# Helper function for insert to recursively find the right place for the node
def _insert(self, root, key):
    if key < root.value:
        if root.left is None:
            root.left = Node(key)
        else:
            self._insert(root.left, key)
    else:
        if root.right is None:
            root.right = Node(key)
        else:
            self._insert(root.right, key)

# Function to count Non-Leaf nodes
def count_non_leaf_nodes(self):
    return self._count_non_leaf_nodes(self.root)

# Helper function for count_non_leaf_nodes to recursively count non-leaf nodes
def _count_non_leaf_nodes(self, node):
    # Base case: If node is None, return 0
    if node is None:
        return 0

    # If node is a leaf node (both left and right are None), return 0
```

```
        if node.left is None and node.right is None:
            return 0

        # Count this node as a non-leaf node and recursively count in both subtrees
        return 1 + self._count_non_leaf_nodes(node.left) + self._count_non_leaf_nodes(node.right)

# Driver code
if __name__ == "__main__":
    # Create the binary search tree
    bst = BinarySearchTree()

    # Insert nodes
    nodes = [50, 30, 20, 40, 70, 60, 80]
    for node in nodes:
        bst.insert(node)

    # Print the count of non-leaf nodes
    print("Number of Non-Leaf nodes:", bst.count_non_leaf_nodes())
```

Q.2 Write a Python program that accepts the vertices and edges of a directed graph. Create and display adjacency list. Implement functions to print out-degree of all vertices of graph.
[20 Mks]

⇨ class Graph:

```
    def __init__(self):
        # Initialize an empty adjacency list
        self.adj_list = {}

    # Function to add a vertex to the graph
    def add_vertex(self, vertex):
        if vertex not in self.adj_list:
            self.adj_list[vertex] = []
```

```
    # Function to add an edge from vertex 'u' to vertex 'v'
    def add_edge(self, u, v):
        if u not in self.adj_list:
            self.add_vertex(u)
        if v not in self.adj_list:
            self.add_vertex(v)
        # Add the destination vertex 'v' to the adjacency list of 'u'
        self.adj_list[u].append(v)

    # Function to display the adjacency list of the graph
    def display_adjacency_list(self):
        print("Adjacency List:")
        for vertex in self.adj_list:
            print(f"{vertex} -> {self.adj_list[vertex]}")

    # Function to calculate the out-degree of all vertices
    def out_degree(self):
        print("\nOut-degree of each vertex:")
        for vertex in self.adj_list:
            out_degree = len(self.adj_list[vertex])
            print(f"Out-degree of {vertex}: {out_degree}")

# Driver Code
if __name__ == "__main__":
    # Create a graph object
    graph = Graph()

    # Accept the number of vertices and edges
    vertices = int(input("Enter number of vertices: "))
    edges = int(input("Enter number of edges: "))
```

```
    # Accept the edges
    print("Enter the edges (u v) where u -> v:")
    for _ in range(edges):
        u, v = map(int, input().split())
        graph.add_edge(u, v)

    # Display the adjacency list
    graph.display_adjacency_list()

    # Print out-degree of all vertices
    graph.out_degree()
```

Q.3 Viva [5 Mks]