SLIP 1

Q.1) Write a Python Program to Calculate the Average of Numbers in a List.

Answer:

```
def average(lst):
    if len(lst) == 0:
        return 0
    s = 0
    for i in lst:
        s = s + i
    return s/len(lst)

nums = [10, 20, 30, 40, 50]
print("Average:", average(nums))
```

Q.2) Write a python program to perform following operations on BST.

Insert
Display

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)

    def display(self):
        self.inorder(self.root)

bst = BST()
vals = [50, 30, 70, 20, 40, 60, 80]
for v in vals:
    bst.insert(v)

bst.display()
```

OR

Q.2) Python program to merge two sorted linked lists.

Answer:

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

def build_list(arr):
    head = None
    temp = None
    for x in arr:
        n = Node(x)
        if head is None:
            head = n
            temp = n
        else:
            temp.next = n
            temp = n
    return head

def merge(l1, l2):
    d = Node(0)
    t = d
    a = l1
    b = l2
    while a and b:
        if a.data <= b.data:
            t.next = Node(a.data)
            a = a.next
        else:
            t.next = Node(b.data)
            b = b.next
        t = t.next
    while a:
        t.next = Node(a.data)
        a = a.next
        t = t.next
    while b:
        t.next = Node(b.data)
        b = b.next
        t = t.next
    return d.next

def display(head):
    t = head
    while t:
        print(t.data, end=" ")
        t = t.next

l1 = build_list([1, 3, 5, 7])
l2 = build_list([2, 4, 6, 8])
m = merge(l1, l2)
display(m)
```

SLIP 2

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”.**

Answer:

```
nums = []
for i in range(6):
    n = int(input("Enter number: "))
    nums.append(n)

if len(nums) != len(set(nums)):
    print("DUPLICATES")
else:
    print("ALL UNIQUE")
```

Q.2) **Write a python program to perform following operations on BST.**

Create
Search
Display (Preorder / Inorder / Postorder)

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def search(self, key):
        cur = self.root
        while cur:
            if key == cur.key:
                return True
            elif key < cur.key:
                cur = cur.left
            else:
                cur = cur.right
        return False

    def preorder(self, node):
        if node:
            print(node.key, end=" ")
            self.preorder(node.left)
            self.preorder(node.right)

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)

    def postorder(self, node):
        if node:
            self.postorder(node.left)
            self.postorder(node.right)
            print(node.key, end=" ")

bst = BST()
vals = [50, 30, 70, 20, 40, 60, 80]
for v in vals:
    bst.insert(v)

print("Search 40:", bst.search(40))
print("Inorder:")
bst.inorder(bst.root)
print("\nPreorder:")
bst.preorder(bst.root)
print("\nPostorder:")
bst.postorder(bst.root)
```

OR

Q.2)

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

Answer:

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

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

    def insert(self, data):
        n = Node(data)
        if self.head is None:
            self.head = n
        else:
            t = self.head
            while t.next:
                t = t.next
            t.next = n

    def display(self):
        t = self.head
        while t:
            print(t.data, end=" ")
            t = t.next

s = SLL()
s.insert(10)
s.insert(20)
s.insert(30)
s.display()
```

SLIP 3

Q.1) Write a Python program to add and remove operation on set.

Answer:

```
s = set()

s.add(10)
s.add(20)
s.add(30)

print("After Add:", s)

s.remove(20)

print("After Remove:", s)
```

Q.2) Write a python program to perform following operations on Binary Search Tree

i. Create
ii. Count non-leaf nodes
iii. Traversal (Preorder / Inorder / Postorder)

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def count_non_leaf(self, node):
        if node is None:
            return 0
        if node.left is None and node.right is None:
            return 0
        return 1 + self.count_non_leaf(node.left) + self.count_non_leaf(node.right)

    def preorder(self, node):
        if node:
            print(node.key, end=" ")
            self.preorder(node.left)
            self.preorder(node.right)

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)

    def postorder(self, node):
        if node:
            self.postorder(node.left)
            self.postorder(node.right)
            print(node.key, end=" ")

bst = BST()
vals = [50, 30, 70, 20, 40, 60, 80]
for v in vals:
    bst.insert(v)

print("Non-Leaf Nodes:", bst.count_non_leaf(bst.root))
print("Inorder:")
bst.inorder(bst.root)
print("\nPreorder:")
bst.preorder(bst.root)
print("\nPostorder:")
bst.postorder(bst.root)
```

OR

Q.2) **Python program for dynamic implementation of Singly Linked List to perform Insert and Display operations.**

Answer:

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

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

    def insert(self, data):
        n = Node(data)
        if self.head is None:
            self.head = n
        else:
            t = self.head
            while t.next:
                t = t.next
            t.next = n

    def display(self):
        t = self.head
        while t:
            print(t.data, end=" ")
            t = t.next

s = SLL()
s.insert(5)
s.insert(15)
s.insert(25)
s.display()
```

SLIP 4

Q.1) **Write a Python program to find maximum and the minimum value in a set.**

Answer:

```
s = {10, 25, 5, 40, 18}

print("Maximum:", max(s))
print("Minimum:", min(s))
```

Q.2) **Write a python program to perform following operations on Binary Search Tree**

i. Create
ii. Count leaf nodes
iii. Traversal (Preorder / Inorder / Postorder)

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def count_leaf(self, node):
        if node is None:
            return 0
        if node.left is None and node.right is None:
            return 1
        return self.count_leaf(node.left) + self.count_leaf(node.right)

    def preorder(self, node):
        if node:
            print(node.key, end=" ")
            self.preorder(node.left)
            self.preorder(node.right)

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)

    def postorder(self, node):
        if node:
            self.postorder(node.left)
            self.postorder(node.right)
            print(node.key, end=" ")

bst = BST()
vals = [50, 30, 70, 20, 40, 60, 80]
for v in vals:
    bst.insert(v)

print("Leaf Nodes:", bst.count_leaf(bst.root))
print("Inorder:")
bst.inorder(bst.root)
print("\nPreorder:")
bst.preorder(bst.root)
print("\nPostorder:")
bst.postorder(bst.root)
```

OR

Q.2) Python program to create a linked list in the sorted order.

Answer:

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

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

    def insert(self, data):
        n = Node(data)
        if self.head is None or data < self.head.data:
            n.next = self.head
            self.head = n
            return
        t = self.head
        while t.next and t.next.data < data:
            t = t.next
        n.next = t.next
        t.next = n

    def display(self):
        t = self.head
        while t:
            print(t.data, end=" ")
            t = t.next
```

```
s = SortedList()
s.insert(30)
s.insert(10)
s.insert(20)
s.insert(40)
s.display()
```

SLIP 5

Q.1) Write a python program to create an array of ‘n’ integers and display the array elements. Access individual elements through indexes.

Answer:

```
n = int(input("Enter size: "))
arr = []

for i in range(n):
    val = int(input("Enter element: "))
    arr.append(val)

print("Array:", arr)

print("Access using index:")
for i in range(n):
    print(arr[i])
```

Q.2) Write a python program to perform following operations on BST

i. Create
ii. Delete
iii. Traversal (Preorder / Inorder / Postorder)

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def delete(self, node, key):
        if node is None:
            return node
        if key < node.key:
            node.left = self.delete(node.left, key)
        elif key > node.key:
            node.right = self.delete(node.right, key)
        else:
            if node.left is None:
                return node.right
            if node.right is None:
                return node.left
            t = node.right
            while t.left:
                t = t.left
            node.key = t.key
            node.right = self.delete(node.right, t.key)
        return node

    def preorder(self, node):
        if node:
            print(node.key, end=" ")
            self.preorder(node.left)
            self.preorder(node.right)

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)

    def postorder(self, node):
        if node:
            self.postorder(node.left)
            self.postorder(node.right)
            print(node.key, end=" ")

bst = BST()
vals = [50, 30, 70, 20, 40, 60, 80]
for v in vals:
    bst.insert(v)

print("Before Delete:")
bst.inorder(bst.root)
bst.root = bst.delete(bst.root, 30)
print("\nAfter Delete:")
bst.inorder(bst.root)
print("\nPreorder:")
bst.preorder(bst.root)
print("\nPostorder:")
bst.postorder(bst.root)
```

OR

Q.2) **Write a python program for implementation of Doubly Linked List to perform Insert and Display operations.**

Answer:

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

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

    def insert(self, data):
        n = Node(data)
        if self.head is None:
            self.head = n
        else:
            t = self.head
            while t.next:
                t = t.next
            t.next = n
            n.prev = t

    def display(self):
        t = self.head
        while t:
            print(t.data, end=" ")
            t = t.next

d = DLL()
d.insert(10)
d.insert(20)
d.insert(30)
d.display()
```

SLIP 6

Q.1) **Write a python program to get the number of occurrences of specified elements in an array.**

Answer:

```
arr = [10, 20, 10, 30, 20, 10]
x = int(input("Enter element to count: "))

count = 0
for i in arr:
    if i == x:
        count += 1

print("Occurrences:", count)
```

Q.2) **Write a python program to perform following operations on Binary Search Tree**

i. Create
ii. Count total nodes
iii. Traversal (Preorder / Inorder / Postorder)

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def count_nodes(self, node):
        if node is None:
            return 0
        return 1 + self.count_nodes(node.left) + self.count_nodes(node.right)

    def preorder(self, node):
        if node:
            print(node.key, end=" ")
            self.preorder(node.left)
            self.preorder(node.right)

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)

    def postorder(self, node):
        if node:
            self.postorder(node.left)
            self.postorder(node.right)
            print(node.key, end=" ")

bst = BST()
vals = [50, 30, 70, 20, 40, 60, 80]
for v in vals:
    bst.insert(v)

print("Total Nodes:", bst.count_nodes(bst.root))
print("Inorder:")
bst.inorder(bst.root)
print("\nPreorder:")
bst.preorder(bst.root)
print("\nPostorder:")
bst.postorder(bst.root)
```

OR

Q.2) **Python program to create doubly linked list and search the given node in the Linked list.**

Answer:

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

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

    def insert(self, data):
        n = Node(data)
        if self.head is None:
            self.head = n
        else:
            t = self.head
            while t.next:
                t = t.next
            t.next = n
            n.prev = t

    def search(self, x):
        t = self.head
        while t:
            if t.data == x:
                return True
            t = t.next
        return False

    def display(self):
        t = self.head
        while t:
            print(t.data, end=" ")
            t = t.next

d = DLL()
d.insert(10)
d.insert(20)
d.insert(30)
d.display()

print("\nSearch 20:", d.search(20))
```

SLIP 7

Q.1) Write a python program to reverse the order of the items in the array.

Answer:

```
arr = [10, 20, 30, 40, 50]

rev = []
for i in range(len(arr)-1, -1, -1):
    rev.append(arr[i])

print("Reversed Array:", rev)
```

Q.2) Write a python program to perform following operations on BST.

i. Create
ii. Display

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)
```

```
bst = BST()
vals = [40, 20, 60, 10, 30, 50, 70]
for v in vals:
    bst.insert(v)

bst.inorder(bst.root)
```

OR

Q.2) Python program to create singly linked list and search the given node in the Linked list.

Answer:

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

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

    def insert(self, data):
        n = Node(data)
        if self.head is None:
            self.head = n
        else:
            t = self.head
            while t.next:
                t = t.next
            t.next = n

    def search(self, x):
        t = self.head
        while t:
            if t.data == x:
                return True
            t = t.next
        return False

    def display(self):
        t = self.head
        while t:
            print(t.data, end=" ")
            t = t.next

s = SLL()
s.insert(5)
s.insert(10)
s.insert(15)
s.display()

print("\nSearch 10:", s.search(10))
```

SLIP 8

Q.1) Write a python program to find sum of all the elements in a list.

Answer:

```
lst = [10, 20, 30, 40, 50]

s = 0
for i in lst:
    s = s + i

print("Sum:", s)
```

Q.2) Write a python program to perform following operations on BST.

i. Insert
ii. Delete
iii. Display (Preorder / Inorder / Postorder)

Answer:

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        if self.root is None:
            self.root = Node(key)
            return
        cur = self.root
        while True:
            if key < cur.key:
                if cur.left is None:
                    cur.left = Node(key)
                    return
                cur = cur.left
            else:
                if cur.right is None:
                    cur.right = Node(key)
                    return
                cur = cur.right

    def delete(self, node, key):
        if node is None:
            return node
        if key < node.key:
            node.left = self.delete(node.left, key)
        elif key > node.key:
            node.right = self.delete(node.right, key)
        else:
            if node.left is None:
                return node.right
            if node.right is None:
                return node.left
            temp = node.right
            while temp.left:
                temp = temp.left
            node.key = temp.key
            node.right = self.delete(node.right, temp.key)
        return node

    def preorder(self, node):
        if node:
            print(node.key, end=" ")
            self.preorder(node.left)
            self.preorder(node.right)

    def inorder(self, node):
        if node:
            self.inorder(node.left)
            print(node.key, end=" ")
            self.inorder(node.right)

    def postorder(self, node):
        if node:
            self.postorder(node.left)
            self.postorder(node.right)
            print(node.key, end=" ")

bst = BST()
vals = [40, 20, 60, 10, 30, 50, 70]
for v in vals:
    bst.insert(v)

print("Inorder before delete:")
bst.inorder(bst.root)
bst.root = bst.delete(bst.root, 20)
print("\nInorder after delete:")
bst.inorder(bst.root)
print("\nPreorder:")
bst.preorder(bst.root)
print("\nPostorder:")
bst.postorder(bst.root)
```

OR

Q.2) **Python program to create singly linked list and reverse the Linked list.**

Answer:

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

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

    def insert(self, data):
        n = Node(data)
        if self.head is None:
            self.head = n
        else:
            t = self.head
            while t.next:
                t = t.next
            t.next = n

    def reverse(self):
        prev = None
        cur = self.head
        while cur:
            nxt = cur.next
            cur.next = prev
            prev = cur
            cur = nxt
        self.head = prev

    def display(self):
        t = self.head
        while t:
            print(t.data, end=" ")
            t = t.next

s = SLL()
s.insert(10)
s.insert(20)
s.insert(30)

print("Original:")
s.display()

s.reverse()
print("\nReversed:")
s.display()
```

SLIP 9

Q.1) **Write a python function to calculate the factorial of a number. The function accepts the number as an argument.**

Answer:

```
def factorial(n):
    f = 1
    for i in range(1, n+1):
        f = f * i
    return f

num = int(input("Enter number: "))
print("Factorial:", factorial(num))
```

Q.2) **Write a program to search an element using Linear Search.**

Answer:

```
arr = [10, 20, 30, 40, 50]
x = int(input("Enter element to search: "))

found = False
for i in range(len(arr)):
    if arr[i] == x:
        found = True
        print("Element found at index", i)
        break

if not found:
    print("Element not found")
```

OR

Q.2) **Write a program to calculate indegree of a graph using adjacency matrix.**

Answer:

```
adj = [
    [0,1,1,0],
    [0,0,1,1],
    [1,0,0,0],
    [0,0,1,0]
]

n = len(adj)
for col in range(n):
    indeg = 0
    for row in range(n):
        indeg += adj[row][col]
    print("Indegree of node", col, "=", indeg)
```

SLIP 10

Q.1) **Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x : x*x).**

```
n = int(input("Enter n: "))
d = {}

for x in range(1, n+1):
    d[x] = x*x

print(d)
```

Q.2) **Write a program to search an element using Binary Search.**

```
def binary_search(arr, x):
    l = 0
    r = len(arr) - 1
    while l <= r:
        m = (l + r) // 2
        if arr[m] == x:
            return m
        elif arr[m] < x:
            l = m + 1
        else:
            r = m - 1
    return -1

arr = [10, 20, 30, 40, 50, 60]
x = int(input("Enter element to search: "))

res = binary_search(arr, x)
if res != -1:
    print("Element found at index", res)
else:
    print("Element not found")
```

OR

Q.2) **Write a Python program to calculate outdegree of a graph using adjacency matrix.**

```
adj = [
    [0,1,1,0],
    [0,0,1,1],
    [1,0,0,0],
    [0,0,1,0]
]

n = len(adj)
for row in range(n):
    outdeg = 0
    for col in range(n):
        outdeg += adj[row][col]
    print("Outdegree of node", row, "=", outdeg)
```

SLIP 11

Q.1) Write a program to generate Fibonacci numbers using function.

Answer:

```
def fib(n):
    a = 0
    b = 1
    for i in range(n):
        print(a, end=" ")
        c = a + b
        a = b
        b = c

num = int(input("Enter how many terms: "))
fib(num)
```

Q.2) Write a Python program to sort given numbers using Bubble Sort algorithm.

Answer:

```
arr = [40, 10, 30, 20, 50]

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]

print("Sorted Array:", arr)
```

OR

Q.2) Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

Answer:

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

    def area(self):
        return 3.14 * self.r * self.r

    def perimeter(self):
        return 2 * 3.14 * self.r
```

```
c = Circle(5)
print("Area:", c.area())
print("Perimeter:", c.perimeter())
```

SLIP 12

Q.1) **Write a Python script to generate and print a dictionary that contains a number (Between 1 and n) in the form (x, x*x).**
Sample Dictionary (n = 5)
Expected Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}.

Answer:

```
n = int(input("Enter n: "))
d = {}

for x in range(1, n+1):
    d[x] = x*x

print(d)
```

Q.2) **Write a Python program to implement sorting Merge Sort algorithm.**

Answer:

```
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

arr = [40, 10, 30, 50, 20]
merge_sort(arr)
print("Sorted Array:", arr)
```

OR

Q.2) **Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.**

Answer:

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

    def add_item(self, name, price):
        self.items[name] = price

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

    def total(self):
        t = 0
        for p in self.items.values():
            t += p
        return t

cart = ShoppingCart()
cart.add_item("Pen", 10)
cart.add_item("Book", 50)
cart.remove_item("Pen")
print("Total Price:", cart.total())
```

SLIP 13

Q.1) Write a Python script to sort (ascending and descending) a dictionary by value.

Answer:

```
d = {"a": 30, "b": 10, "c": 20}

asc = dict(sorted(d.items(), key=lambda x: x[1]))
desc = dict(sorted(d.items(), key=lambda x: x[1], reverse=True))

print("Ascending:", asc)
print("Descending:", desc)
```

Q.2) Write a Python program to search an element in an integer array using Binary Search.

Answer:

```
def binary_search(arr, x):
    l = 0
    r = len(arr) - 1
    while l <= r:
        m = (l + r) // 2
        if arr[m] == x:
            return m
        elif arr[m] < x:
            l = m + 1
        else:
            r = m - 1
    return -1

arr = [5, 10, 15, 20, 25, 30]
x = int(input("Enter element to search: "))

res = binary_search(arr, x)
if res != -1:
    print("Element found at index", res)
else:
    print("Element not found")
```

OR

Q.2) Write a Python program to implement sorting Quick Sort algorithm.

Answer:

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[0]
    left = []
    right = []
    for x in arr[1:]:
        if x < pivot:
            left.append(x)
        else:
            right.append(x)
    return quick_sort(left) + [pivot] + quick_sort(right)

arr = [40, 10, 30, 20, 50]
arr = quick_sort(arr)
print("Sorted Array:", arr)
```

SLIP 14

Q.1) Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x : x*x).

Answer:

```
n = int(input("Enter n: "))
d = {}

for x in range(1, n+1):
    d[x] = x*x

print(d)
```

Q.2) Write a Python program to implement sorting Insertion Sort algorithm.

Answer:

```
arr = [40, 10, 30, 20, 50]

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

print("Sorted Array:", arr)
```

OR

Q.2) Write a program to calculate indegree of a graph.

Answer:

```
adj = [
    [0,1,0,1],
    [0,0,1,0],
    [1,0,0,0],
    [0,1,1,0]
]

n = len(adj)
for col in range(n):
    indeg = 0
    for row in range(n):
        indeg += adj[row][col]
    print("Indegree of node", col, "=", indeg)
```

SLIP 15

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

Sample Dictionary:
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})

Answer:

```
d1 = {'a':100,'b':200,'c':300}
d2 = {'a':300,'b':200,'d':400}

d = {}

for k in d1:
    d[k] = d1[k]

for k in d2:
    if k in d:
        d[k] = d[k] + d2[k]
    else:
        d[k] = d2[k]

print(d)
```

Q.2) **Write a program to calculate outdegree of a graph.**

Answer:

```
adj = [
    [0,1,1,0],
    [0,0,1,1],
    [1,0,0,0],
    [0,0,1,0]
]

n = len(adj)
for row in range(n):
    outdeg = 0
    for col in range(n):
        outdeg += adj[row][col]
    print("Outdegree of node", row, "=", outdeg)
```

OR

Q.2) **Write a Python class named rectangle constructed by a radius and two methods which will compute the area and the perimeter of a circle.**

Answer:

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

    def area(self):
        return 3.14 * self.r * self.r

    def perimeter(self):
        return 2 * 3.14 * self.r

c = Circle(7)
print("Area:", c.area())
print("Perimeter:", c.perimeter())
```

SLIP 16

Q.1) **Write a Python program to create a list of tuples with the first element as the number and second element as the square of the number, also display original list in reverse.**

Answer:

```
lst = [1, 2, 3, 4, 5]

t = []
for x in lst:
    t.append((x, x*x))

print("Tuple List:", t)

print("Reversed Original List:", lst[::-1])
```

Q.2) **Write a python code for static implementation of stack.**

Answer:

```
SIZE = 5
stack = [0] * SIZE
top = -1

def push(x):
    global top
    if top == SIZE-1:
        print("Stack Full")
    else:
        top += 1
        stack[top] = x

def pop():
    global top
    if top == -1:
        print("Stack Empty")
    else:
        val = stack[top]
        top -= 1
        return val

def display():
    if top == -1:
        print("Stack Empty")
    else:
        for i in range(top, -1, -1):
            print(stack[i], end=" ")

push(10)
push(20)
push(30)
display()
print("\nPopped:", pop())
display()
```

OR

Q.2) Write a Python program for Evaluation of postfix expression.

Answer:

```
def evaluate_postfix(exp):
    stack = []
    for ch in exp:
        if ch.isdigit():
            stack.append(int(ch))
        else:
            b = stack.pop()
            a = stack.pop()
            if ch == '+':
                stack.append(a + b)
            elif ch == '-':
                stack.append(a - b)
            elif ch == '*':
                stack.append(a * b)
            elif ch == '/':
                stack.append(a / b)
    return stack.pop()

exp = "23*54*+"
print("Result:", evaluate_postfix(exp))
```

SLIP 17

Q.1) Write a Python Program to Calculate the Average of Numbers in a Given List.

Answer:

```
lst = [10, 20, 30, 40, 50]

s = 0
for x in lst:
    s = s + x

print("Average:", s/len(lst))
```

Q.2) Write a python code for static implementation of queue.

Answer:

```
SIZE = 5
queue = [0] * SIZE
front = -1
rear = -1

def enqueue(x):
    global front, rear
    if rear == SIZE - 1:
        print("Queue Full")
    else:
        if front == -1:
            front = 0
        rear += 1
        queue[rear] = x

def dequeue():
    global front, rear
    if front == -1 or front > rear:
        print("Queue Empty")
    else:
        val = queue[front]
        front += 1
        return val

def display():
    if front == -1 or front > rear:
        print("Queue Empty")
    else:
        for i in range(front, rear+1):
            print(queue[i], end=" ")

enqueue(10)
enqueue(20)
enqueue(30)
display()
print("\nDequeued:", dequeue())
display()
```

OR

Q.2) Write a python code for dynamic implementation of Stack to perform following operations: Init, Push, Pop, Isempty, Isfull.

Answer:

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

    def push(self, x):
        if len(self.stack) == self.size:
            print("Stack Full")
        else:
            self.stack.append(x)

    def pop(self):
        if len(self.stack) == 0:
            print("Stack Empty")
        else:
            return self.stack.pop()

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

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

    def display(self):
        print(self.stack)

s = Stack(5)
s.push(10)
s.push(20)
s.push(30)
s.display()
print("Popped:", s.pop())
s.display()
```

SLIP 18

Q.1) **Write a python code to copy element 44 and 55 from the following tuple into a new tuple tuple1 = (11, 22, 33, 44, 55, 66), also display the same tuple in reverse order.**

Answer:

```
tuple1 = (11, 22, 33, 44, 55, 66)

new_tuple = (tuple1[3], tuple1[4])
print("New Tuple:", new_tuple)

print("Reverse:", tuple1[::-1])
```

Q.2) **Write a python code for simple implementation of priority queue.**

Answer:

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

    def insert(self, value):
        self.q.append(value)
        self.q.sort()

    def delete(self):
        if len(self.q) == 0:
            print("Queue Empty")
        else:
            return self.q.pop(0)

    def display(self):
        print(self.q)

p = PriorityQueue()
p.insert(30)
p.insert(10)
p.insert(20)
p.display()
print("Deleted:", p.delete())
p.display()
```

OR

Q.2) **Write a Python program to convert infix to postfix conversion using stack.**

Answer:

```
precedence = {
    '+':1, '-':1, '*':2, '/':2, '^':3
}

def infix_to_postfix(exp):
    stack = []
    output = ""

    for ch in exp:
        if ch.isalnum():
            output += ch
        elif ch == '(':
            stack.append(ch)
        elif ch == ')':
            while stack and stack[-1] != '(':
                output += stack.pop()
            stack.pop()
        else:
            while stack and stack[-1] != '(' and precedence[ch] <= precedence[stack[-
1]]:
                output += stack.pop()
            stack.append(ch)

    while stack:
        output += stack.pop()

    return output

exp = "A+(B*C-(D/E^F)*G)*H"
print("Postfix:", infix_to_postfix(exp))
```

SLIP 19

Q.1) Write a Python program to get the 5th element from front and 5th element from last of a tuple.

Answer:

```
t = (11, 22, 33, 44, 55, 66, 77, 88, 99, 111)

fifth_front = t[4]
fifth_last = t[-5]

print("5th from front:", fifth_front)
print("5th from last:", fifth_last)
```

Q.2) Write a python code for simple implementation of priority queue.

Answer:

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

    def insert(self, val):
        self.q.append(val)
        self.q.sort()

    def delete(self):
        if len(self.q) == 0:
            print("Queue Empty")
        else:
            return self.q.pop(0)

    def display(self):
        print(self.q)

p = PriorityQueue()
p.insert(40)
p.insert(10)
p.insert(30)
p.display()
print("Deleted:", p.delete())
p.display()
```

OR

Q.2) Write a python code for dynamic implementation of Stack to perform following operations: Init, Push, Pop, Isempty, Isfull.

Answer:

```
class Stack:
    def __init__(self, size):
        self.size = size
        self.st = []
    def push(self, x):
        if len(self.st) == self.size:
            print("Stack Full")
        else:
            self.st.append(x)

    def pop(self):
        if len(self.st) == 0:
            print("Stack Empty")
        else:
            return self.st.pop()

    def isempty(self):
        return len(self.st) == 0

    def isfull(self):
        return len(self.st) == self.size

    def display(self):
        print(self.st)

s = Stack(5)
s.push(10)
s.push(20)
s.push(30)
s.display()
print("Popped:", s.pop())
s.display()
```

SLIP 20

Q.1) Write a program to display following pattern.

```
1
2 3
4 5 6
7 8 9 10
```

Answer:

```
n = 4
num = 1

for i in range(1, n+1):
    for j in range(i):
        print(num, end=" ")
        num += 1
    print()
```

Q.2) Write a code for static stack implementation in python.

Answer:

```
SIZE = 5
stack = [0] * SIZE
top = -1

def push(x):
    global top
    if top == SIZE - 1:
        print("Stack Full")
    else:
        top += 1
        stack[top] = x

def pop():
    global top
    if top == -1:
        print("Stack Empty")
    else:
        val = stack[top]
        top -= 1
        return val

def display():
    if top == -1:
        print("Stack Empty")
    else:
        for i in range(top, -1, -1):
            print(stack[i], end=" ")

push(10)
push(20)
push(30)
display()
print("\nPopped:", pop())
display()
```

OR

Q.2) **Write a python code for dynamic implementation of linear Queue to perform following operations: init, enqueue, dequeue, isEmpty, isFull.**

Answer:

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

    def enqueue(self, x):
        if len(self.q) == self.size:
            print("Queue Full")
        else:
            self.q.append(x)

    def dequeue(self):
        if len(self.q) == 0:
            print("Queue Empty")
        else:
            return self.q.pop(0)

    def isEmpty(self):
        return len(self.q) == 0

    def isFull(self):
        return len(self.q) == self.size

    def display(self):
        print(self.q)

q = Queue(5)
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)
q.display()
print("Dequeued:", q.dequeue())
q.display()
```

SLIP 21

Q.1) **Write a python program which accepts 6 integer values and prints “DUPLICATES” if any of the values entered are duplicates otherwise it prints “ALL UNIQUE”.**
Example: (32, 10, 45, 90, 45, 6) → “DUPLICATES”.

Answer:

```
nums = []
for i in range(6):
    n = int(input("Enter number: "))
    nums.append(n)

if len(nums) == len(set(nums)):
    print("ALL UNIQUE")
else:
    print("DUPLICATES")
```

Q.2) **Show the static implementation of queue using Python.**

Answer:

```
SIZE = 5

queue = [0] * SIZE
front = -1
rear = -1

def enqueue(x):
    global front, rear
    if rear == SIZE - 1:
        print("Queue Full")
    else:
        if front == -1:
            front = 0
        rear += 1
        queue[rear] = x

def dequeue():
    global front, rear
    if front == -1 or front > rear:
        print("Queue Empty")
    else:
        val = queue[front]
        front += 1
        return val

def display():
    if front == -1 or front > rear:
        print("Queue Empty")
    else:
        for i in range(front, rear+1):
            print(queue[i], end=" ")

enqueue(10)
enqueue(20)
enqueue(30)
display()
print("\nDequeued:", dequeue())
display()
```

OR

Q.2) **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.**

Answer:

```
def reverse_string(s):
    stack = []
    for ch in s:
        stack.append(ch)
    rev = ""
    while stack:
        rev += stack.pop()
    return rev

s = input("Enter string: ")
rev = reverse_string(s)

print("Reversed:", rev)

if s == rev:
    print("Palindrome")
else:
    print("Not Palindrome")
```

SLIP 22

Q.1) Write a Python program to find repeated items in a tuple.

Answer:

```
t = (11, 22, 33, 22, 44, 11, 55)

repeated = []
for x in t:
    if t.count(x) > 1 and x not in repeated:
        repeated.append(x)

print("Repeated Items:", repeated)
```

Q.2) Show the static implementation of stack using python.

Answer:

```
SIZE = 5
stack = [0] * SIZE
top = -1

def push(x):
    global top
    if top == SIZE - 1:
        print("Stack Full")
    else:
        top += 1
        stack[top] = x

def pop():
    global top
    if top == -1:
        print("Stack Empty")
    else:
        val = stack[top]
        top -= 1
        return val

def display():
    if top == -1:
        print("Stack Empty")
    else:
        for i in range(top, -1, -1):
            print(stack[i], end=" ")

push(10)
push(20)
push(30)
display()
print("\nPopped:", pop())
display()
```

OR

Q.2)

Write a python code for implementation of circular queue.
[20 Marks]

Answer:

```
SIZE = 5
cq = [0] * SIZE
front = -1
rear = -1

def enqueue(x):
    global front, rear
    if (rear + 1) % SIZE == front:
        print("Queue Full")
    else:
        if front == -1:
            front = 0
        rear = (rear + 1) % SIZE
        cq[rear] = x

def dequeue():
    global front, rear
    if front == -1:
        print("Queue Empty")
    else:
        val = cq[front]
        if front == rear:
            front = rear = -1
        else:
            front = (front + 1) % SIZE
        return val

def display():
    if front == -1:
        print("Queue Empty")
    else:
        i = front
        while True:
            print(cq[i], end=" ")
            if i == rear:
                break
            i = (i + 1) % SIZE

enqueue(10)
enqueue(20)
enqueue(30)
display()
print("\nDequeued:", dequeue())
display()
```

SLIP 23

Q.1) Write a Python Program to Calculate the Average of Numbers in a Given List.

Answer:

```
lst = [10, 20, 30, 40, 50]

s = 0
for x in lst:
    s += x

print("Average:", s/len(lst))
```

Q.2)

Write the code for static queue implementation in Python.

Answer:

```
SIZE = 5
queue = [0] * SIZE
front = -1
rear = -1

def enqueue(x):
    global front, rear
    if rear == SIZE - 1:
        print("Queue Full")
    else:
        if front == -1:
            front = 0
        rear += 1
        queue[rear] = x

def dequeue():
    global front, rear
    if front == -1 or front > rear:
        print("Queue Empty")
    else:
        val = queue[front]
        front += 1
        return val

def display():
    if front == -1 or front > rear:
        print("Queue Empty")
    else:
        for i in range(front, rear + 1):
            print(queue[i], end=" ")

enqueue(10)
enqueue(20)
enqueue(30)
display()
print("\nDequeued:", dequeue())
display()
```

OR

Q.2)

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.

Answer:

```
def reverse_string(s):
    stack = []
    for ch in s:
        stack.append(ch)
    rev = ""
    while stack:
        rev += stack.pop()
    return rev

s = input("Enter string: ")
rev = reverse_string(s)

print("Reversed:", rev)

if s == rev:
    print("Palindrome")
else:
    print("Not Palindrome")
```

SLIP 24

Q.1)

Write a Python Program to Calculate the Sum of Numbers in a List.

Answer:

```
lst = [10, 20, 30, 40, 50]

s = 0
for x in lst:
    s = s + x

print("Sum:", s)
```

Q.2)

Write a program to search an element using Binary Search.

Answer:

```
def binary_search(arr, x):
    l = 0
    r = len(arr) - 1
    while l <= r:
        m = (l + r) // 2
        if arr[m] == x:
            return m
        elif arr[m] < x:
            l = m + 1
        else:
            r = m - 1
    return -1

arr = [10, 20, 30, 40, 50, 60]
x = int(input("Enter element to search: "))

res = binary_search(arr, x)
if res != -1:
    print("Element found at index", res)
else:
    print("Element not found")
```

OR

Q.2)

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

Answer:

```
adj = [
    [0,1,1,0],
    [0,0,1,1],
    [1,0,0,0],
    [0,0,1,0]
]

n = len(adj)

for row in range(n):
    outdeg = 0
    for col in range(n):
        outdeg += adj[row][col]
    print("Outdegree of node", row, "=", outdeg)
```

SLIP 25

Q.1)

Write a python program to get the number of occurrences of specified elements in an array.

Answer:

```
arr = [10, 20, 10, 30, 20, 10]
x = int(input("Enter element to count: "))

count = 0
for i in arr:
    if i == x:
        count += 1

print("Occurrences:", count)
```

Q.2)

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

Answer:

```
arr = [40, 10, 20, 50, 30]

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]

print("Sorted Array:", arr)
```

OR

Q.2)

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

Answer:

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

    def area(self):
        return 3.14 * self.r * self.r

    def perimeter(self):
        return 2 * 3.14 * self.r
```

```
c = Circle(7)
print("Area:", c.area())
print("Perimeter:", c.perimeter())
```