PURANDHAR TECHNICAL EDUCATION SOCIETY'S **PUNE**
CAMBRIDGE INSTITUTE OF MANAGEMENT AND COMPUTER APPLICATION
Affiliated to Savitribai Phule Pune University Recognized by A.I.C.T.E. New Delhi & Govt of Maharashtra ,DTE CODE - 6992

Practical based on Python Programming

Practical Journal

Name of Student:

Course: MCA-I Semester-I

Roll No.: 24040

Subject : [IT-11L]- Practical based on Python and DS

Subject Faculty : Prof. Shubham R. Jambhale

Certificate

This is to certify that **Gawali Onkar Bharat** is a bonafide student of **PTES's, Pune Cambridge Institute of Management & Computer Application, Pune** has successfully completed the Lab Assignment for **Practical based on Python and DS** as prescribed by the **Savitribai Phule Pune University**, in the partial fulfilment of the **MCA-I Year Semester-I** curriculum as per the rules of Savitribai Phule Pune University.

University Exam Seat No: **Roll No.: MC24040**

Prof. Shubham R. Jambhale	**Dr. Rachana Chavan**	**Dr. Renuka Vanarase**
Practical Incharge	**Dean Academics**	**Director, PCIMCA**

External Examiner:

Internal Examiner:

Date: / /2024

Place: Pune

Subject : Python Programming

Sr.No	Title	Page no	Sign
1	Write a python function for the Armstrong number, palindrome number and prime number using switch case.		
2	Using predefined functions perform string operations .		
3	Using input () function accepts customer details with validation (name,email,phone number)		
4	Create a module ‘arithmetic’ which has classes . (substraction , division , addition and multiplication)		
5	Write a program that accepts details of students from user and store in the directory ,store details in ‘student.txt’		
6	Write a program for exception handling with file handling operations		
7	Write a python program that demonstrate various modes to implement file handling operations.		
8	Using hybrid inheritance ,create super class animal and respective derived class.		
9	Demonstrate generators and decorators with suitable examples.		
10	Write a python program which demonstrate difference between search() and find() function in re module.		
11	Using at least 6 special character sequence create matching patterns		
12	Write a python program that create 3 threads and implements synchronisation.		
13	Create a database connectivity which implements validations.		
14	Accepts details of employee (ename ,eid, ..) from user and store in a database ,following operations are required, 1. search using id 2. delete specified employee records 3. find duplicate records.		
15	Using list ,dictionary, tuple store details in database.		

Q.1 Write a python function for the Armstrong number, palindrome number and prime number using switch case

Program:

```
def is_armstrong(num):
    return num == sum(int(digit) ** len(str(num)) for digit in str(num))

def is_palindrome(num):
    return str(num) == str(num)[::-1]

def is_prime(num):
    return num > 1 and all(num % i for i in range(2, int(num ** 0.5) + 1))

def main():
    choice = int(input("Choose 1 for Armstrong, 2 for Palindrome, 3 for Prime: "))
    num = int(input("Enter a number: "))

    if choice == 1:
        print(f"{num} is Armstrong.") if is_armstrong(num) else print(f"{num} is not
Armstrong.")
    elif choice == 2:
        print(f"{num} is Palindrome.") if is_palindrome(num) else print(f"{num} is not
Palindrome.")
    elif choice == 3:
        print(f"{num} is Prime.") if is_prime(num) else print(f"{num} is not Prime.")
    else:
        print("Invalid choice.")

if__name__== "_main_":
    main()
```

Q.2 Using predefined functions perform string operations .

Program:

```
str1 = "Hello World"

print(str1.capitalize())
print(str1.lower())
print(str1.isalnum())
print(str1.count("a"))
print(str1.split())
print(str1.isupper())
print(str1.islower())
```

Q.3 Using input () function accepts customer details with validation (name,email,phone number)

Program:

```
def validate_name(name):
    return name.isalpha() and len(name) > 1

def validate_email(email):
    return "@" in email and "." in email and len(email) > 5

def validate_phone(phone):
    return phone.isdigit() and len(phone) == 10

def get_customer_details():

    name = input("Enter your name: ")
    while not validate_name(name):
        print("Invalid name. Please enter a valid name.")
        name = input("Enter your name: ")

    email = input("Enter your email: ")
    while not validate_email(email):
        print("Invalid email. Please enter a valid email.")
        email = input("Enter your email: ")

    phone = input("Enter your phone number (10 digits): ")
    while not validate_phone(phone):
        print("Invalid phone number. Please enter a valid 10-digit phone number.")
        phone = input("Enter your phone number (10 digits): ")

    print("\nCustomer Details:")
    print(f"Name: {name}")
    print(f"Email: {email}")
    print(f"Phone Number: {phone}")

get_customer_details()
```

Q.4 Create a module ‘arithmetic’ which has classes . (substraction , division , addition and multiplication)

Program:

main.py

```
from arithmetic import Addition , Subtraction , Multiplication , Division

def main():

    add = Addition()
    sub = Subtraction()
    mul = Multiplication()
    div = Division()

    a = float(input("Enter the Number: "))
    b = float(input("Enter the number:"))

    print(f"Addition:{a} + {b} = {add.add(a,b)}")
    print(f"Subtraction:{a} - {b} = {sub.subtract(a,b)}")
    print(f"Multiplication:{a} * {b} = {mul.multiply(a,b)}")
    print(f"Division:{a} / {b} = {div.divide(a,b)}")

main()
```

arithmetic.py

```
class Addition:
    def add(self, a, b):
        return a + b

class Subtraction:
    def subtract(self, a, b):
        return a - b

class Multiplication:
    def multiply(self, a, b):
        return a * b

class Division:
    def divide(self, a, b):
        return a / b
```

Q.5 Write a program that accepts details of students from user and store in the directory ,store details in ‘student.txt’

Program:

```
def main():
    name = input("Enter student's name: ")
    age = input("Enter student's age: ")
    grade = input("Enter student's grade: ")

    with open("student.txt", "a") as file:
        file.write(f"{name}, {age}, {grade}\n")

    print("Student details saved.")

if__name__== "_main_":
    main()
```

Student .txt

Ram , 21 , A
Sham , 31 , B
Anup ,24 , C

Q. 6 Write a program for exception handling with file handling operations

Programm:

```
def main():
    try:
        with open("example.txt", "w") as file:
            file.write("Hello, this is a test file.\n")
            file.write("Handling file exceptions easily!\n")
        print("Data written to the file successfully.")

        with open("example.txt", "r") as file:
            content = file.read()
            print("File content:")
            print(content)

    except FileNotFoundError:
        print("The file was not found.")
    except IOError:
        print("An error occurred while handling the file.")
    except Exception as e:
        print(f"Unexpected error: {e}")


main()
```

Q.7 Write a python program that demonstrate various modes to implement file handling operations.

Program:

```
def demonstrate_file_modes():
    with open("example.txt", "w") as file:
        file.write("I am a MCA Student\n")

    with open("example.txt", "a") as file:
        file.write("Doing practical exam \n")

    with open("example.txt", "r") as file:
        print("File content in 'r' mode:")
        print(file.read())

    with open("example.bin", "wb") as file:
        file.write(b"Binary data will  inlcude")

demonstrate_file_modes()
```

example.txt

File content in 'r' mode:
I am a MCA Student
Doing practical exam

Q.8 Using hybrid inheritance ,create super class animal and respective derived class.

Program:

```
class Animal:
    def __init_(self, name):
        self.name = name

    def speak(self):
        print(f"{self.name} makes a sound.")

class Mammal(Animal):
    def __init_(self, name, warm_blooded):
        Animal._init_(self, name)  # Explicit call to Animal's constructor
        self.warm_blooded = warm_blooded

    def info(self):
        print(f"{self.name} is a mammal. Warm-blooded: {self.warm_blooded}")

class Bird(Animal):
    def __init_(self, name, can_fly):
        Animal._init_(self, name)  # Explicit call to Animal's constructor
        self.can_fly = can_fly

    def info(self):
        print(f"{self.name} is a bird. Can fly: {self.can_fly}")

class Bat(Mammal, Bird):
    def __init_(self, name, warm_blooded, can_fly):
        Mammal._init_(self, name, warm_blooded)  # Call Mammal's constructor
        Bird._init_(self, name, can_fly)  # Call Bird's constructor

    def info(self):
        print(f"{self.name} is a bat. Warm-blooded: {self.warm_blooded}, Can fly: {self.can_fly}")

def main():
    bat = Bat("Bat", True, True)  # Creating the Bat object
    bat.info()  # Displaying info

main()
```

Output:

Bat is a bat.Warm-blooded:True,Can fly:True

Q.9 Demonstrate generators and decorators with suitable example.

Program:

```
# Generator function

def square_numbers(n):
    for i in range(n):
        yield i * i

# object
gen = square_numbers(5)


for square in gen:
    print(square)




# Decorator Function

def decor(printer):
    def inner():
        printer()
        print("Tech world")
    return inner

@decor
def printer():
     print("Welcome! ")
     print("to")

printer()
```

Q.10 Write a python program which demonstrates the difference between search( ) and find ( ) function In re module.

Program:

```
import re

text = "Hello, welcome to Python."

search_result = re.search(r"welcome", text)
print("re.search() found:", search_result.group() if search_result else "Not found")

find_result = text.find("welcome")
print("str.find() found at index:", find_result if find_result != -1 else "Not found")
```

Q.11 Using at least 6 special characters sequence create matching patterns.

Program:

```
import re
pattern = r"b.t"  # . means any character can be in place of "t"
text = "bat"
result = re.match(pattern, text)
print(result.group())  # Output: bat

pattern = r"^Hello"  # ^ means the word must start with "Hello"
text = "Hello World"
result = re.match(pattern, text)
print(result.group())  # Output: Hello

pattern = r"world$"  # $ means the word must end with "world"
text = "Hello world"
result = re.search(pattern, text)
print(result.group())  # Output: world

pattern = r"ba*t"  # * means "a" can appear 0 or more times
text = "bt"
result = re.match(pattern, text)
print(result.group())  # Output: bt

pattern = r"ba+t"  # + means "a" must appear 1 or more times
text = "baat"
result = re.match(pattern, text)
print(result.group())  # Output: baat

pattern = r"[bB]at"  # [bB] means match either 'b' or 'B'
text = "Bat"
result = re.match(pattern, text)
print(result.group())  # Output: Bat

patterns = [
   (r"b.t", "bat"),     # Dot
   (r"^Hello", "Hello"), # Caret
   (r"world$", "world"),  # Dollar
   (r"ba*t", "bt"),      # Asterisk
   (r"ba+t", "baat"),    # Plus
   (r"[bB]at", "Bat")    # Square Brackets
]

for pattern, text in patterns:
   result = re.match(pattern, text)
   print(f"Pattern: {pattern}, Text: {text} -> Match:", result.group() if result else "No match")
```

Q.12 Write python program that create 3 threads and implements synchronization.

Program:

```
import threading

counter = 0
lock = threading.Lock() # Lock to synchronize threads

def increment():# Function to increment counter
    global counter
    with lock:
        counter += 1

threads = [threading.Thread(target=increment) for _ in range(3)] # Create 3 threads


# Start threads
for thread in threads:
    thread.start()

# Wait for all threads to finish
for thread in threads:
    thread.join()

print("Final counter value:", counter)
```

Output:

```
Final counter value: 3
```

Q.13 Create a database connectivity which implements validations.

Program:

```
import

conn = sqlite3.connect('students.db')
cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS student (
                id INTEGER PRIMARY KEY,
                name TEXT,
                email TEXT
              )''')

def validate_name(name):
  return len(name) >= 2

def validate_email(email):
   return '@' in email and '.' in email

def insert_student(name, email):
   if not validate_name(name):
      print("Invalid name! Name must be at least 2 characters.")
      return
   if not validate_email(email):
      print("Invalid email format!")
      return

   cursor.execute("INSERT INTO student (name, email) VALUES (?, ?)", (name, email))
   conn.commit()
   print("Student added successfully!")

name = input("Enter student name: ")
email = input("Enter student email: ")

insert_student(name, email)

conn.close()
```

Q.14 Accepts details of employee (ename ,eid, ..) from user and store in a database ,following operations are required,

1. search using id
2. delete specified employee records
3. find duplicate records.

Program:

```
import mysql.connector

def connect_db():
   return mysql.connector.connect(host="localhost", user="root", password="password",
database="employee_db")

def create_table():
   with connect_db() as conn:
      cursor = conn.cursor()
      cursor.execute('''CREATE TABLE IF NOT EXISTS employees (eid INT PRIMARY KEY,
ename VARCHAR(255), department VARCHAR(100), salary FLOAT)''')
      conn.commit()

def insert_employee(eid, ename, department, salary):
   with connect_db() as conn:
      cursor = conn.cursor()
      cursor.execute("INSERT INTO employees (eid, ename, department, salary) VALUES (%s, %s,
%s, %s)", (eid, ename, department, salary))
      conn.commit()

def search_employee(eid):
   with connect_db() as conn:
      cursor = conn.cursor()
      cursor.execute("SELECT * FROM employees WHERE eid = %s", (eid,))
      result = cursor.fetchone()
      print(result if result else "Employee not found.")

def delete_employee(eid):
   with connect_db() as conn:
      cursor = conn.cursor()
      cursor.execute("DELETE FROM employees WHERE eid = %s", (eid,))
      conn.commit()

def find_duplicates():
   with connect_db() as conn:
      cursor = conn.cursor()
      cursor.execute("SELECT eid, COUNT(*) FROM employees GROUP BY eid HAVING
COUNT(*) > 1")
      duplicates = cursor.fetchall()
      print(duplicates if duplicates else "No duplicates found.")
```

```
# Main menu

def main():
    create_table()
    while True:
        choice = input("1. Add\n2. Search\n3. Delete\n4. Find Duplicates\n5. Exit\nChoose: ")

        if choice == '1':
            eid = int(input("ID: "))
            ename = input("Name: ")
            dept = input("Department: ")
            salary = float(input("Salary: "))
            insert_employee(eid, ename, dept, salary)

        elif choice == '2':
            eid = int(input("Enter ID to search: "))
            search_employee(eid)

        elif choice == '3':
            eid = int(input("Enter ID to delete: "))
            delete_employee(eid)

        elif choice == '4':
            find_duplicates()

        elif choice == '5':
            break

if __name__ == "__main__":
    main()
```

Q.15 Using list ,dictionary, tuple store details in database

Program:

```
import mysql.connector

# Connect to MySQL
def connect_db():
    return mysql.connector.connect(host="localhost", user="root", password="password",
database="employee_db")

# Create table
def create_table():
    with connect_db() as conn:
        conn.cursor().execute('''CREATE TABLE IF NOT EXISTS employees (eid INT PRIMARY
KEY, ename VARCHAR(255), department VARCHAR(100), salary FLOAT)''')
        conn.commit()


def insert_employee(eid, ename, dept, salary):
    employee = {'eid': eid, 'ename': ename, 'department': dept, 'salary': salary}
    with connect_db() as conn:
        conn.cursor().execute("INSERT INTO employees (eid, ename, department, salary) VALUES (%s,
%s, %s, %s)",
                              (employee['eid'], employee['ename'], employee['department'], employee['salary']))
        conn.commit()


def search_employee(eid, data):
    for emp in data:
        if emp[0] == eid:
            print(emp)
            return
    print("Employee not found.")
```

```
def main():
    create_table()
    employee_list = [(101, "John", "HR", 50000), (102, "Alice", "IT", 60000)]
    employee_tuple = (101, "John", "HR", 50000)

    while True:
        choice = input("1. Add Employee\n2. Search Employee List\n3. Search Employee Tuple\n4. Exit\nChoice: ")
        if choice == '1':
            eid, ename, dept, salary = int(input("ID: ")), input("Name: "), input("Department: "), float(input("Salary: ")))
            insert_employee(eid, ename, dept, salary)
        elif choice == '2':
            search_employee(int(input("Enter ID to search (List): ")), employee_list)
        elif choice == '3':
            search_employee(int(input("Enter ID to search (Tuple): ")), [employee_tuple])
        else: break

if __name__ == "_main_":
    main()
```

Subject : Data Structure and Algorithms

INDEX

Q.1 Write python program to insert delete and display arrayelements

Program:

```
import array

class ArrayOperations:
    def __init__(self):
        self.array = []

    def insert(self, element):
        self.array.append(element)
        print(f'Element {element} inserted.')

    def delete(self, element):
        if element in self.array:
            self.array.remove(element)
            print(f'Element {element} deleted.')
        else:
            print(f'Element {element} not found in the array.')

    def display(self):
        if self.array:
            print('Array elements:', self.array)
        else:
            print('Array is empty.')

arr_ops = ArrayOperations()

arr_ops.insert(10)
arr_ops.insert(20)
arr_ops.insert(30)

arr_ops.display()

arr_ops.delete(20)

arr_ops.display()
```

Q.2 Write program to convert a matrix to sparse matrix

Program:

```
def to_sparse_matrix(matrix):
    sparse_matrix = []
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] != 0:
                sparse_matrix.append((i, j, matrix[i][j]))
    return sparse_matrix

# Example usage:
matrix = [
    [0, 0, 3],
    [4, 0, 0],
    [0, 0, 5]
]

sparse_matrix = to_sparse_matrix(matrix)
print("Sparse Matrix:", sparse_matrix)
```

Output:

Sparse Matrix: [(0, 2, 3), (1, 0, 4), (2, 3, 5), (3, 1, 2)]

Q.3 Write Python program to demonstrate Implementing stackusing list

Program:

```
class Stack:
    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()
        else:
            return "Stack is empty"

    def peek(self):
        if not self.is_empty():
            return self.stack[-1]
        else:
            return "Stack is empty"

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

# Example usage:
stack = Stack()
stack.push(10)
stack.push(20)
stack.push(30)

print("Top element:", stack.peek())  # Output: 30
print("Popped element:", stack.pop())  # Output: 30
print("Top element after pop:", stack.peek())  # Output: 20
```

Output:

```
Top element: 30
Popped element: 30
Top element after pop: 20
```

Q.4 write a program to find Sum of Array

Program:

```
def sum_of_array(arr):
    return sum(arr)

# Example usage:

array = [1, 2, 3, 4, 5]
result = sum_of_array(array)
print("Sum of array:", result)
```

Output:
Sum of array: 15

Q.5 write an array to find second maximum number of an array

Program:

```
def second_max(arr):
    arr = list(set(arr))
    arr.sort()
    return arr[-2] if len(arr) > 1 else None

arr = [12, 35, 1, 10, 34, 1]
print("Second maximum number is:", second_max(arr))
```

Output:
Second maximum number is: 34

Q.6 Write a program to create doubly linked list for insert delete search print operation using menu driven program

Program:

```
class Node:
    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 = Node(data)
        if not self.head:
            self.head = new_node
        else:
            temp = self.head
            while temp.next:
                temp = temp.next
            temp.next = new_node
            new_node.prev = temp

    def delete(self, data):
        temp = self.head
        while temp and temp.data != data:
            temp = temp.next
        if temp:
            if temp.prev:
                temp.prev.next = temp.next
            if temp.next:
                temp.next.prev = temp.prev
            if temp == self.head:
                self.head = temp.next
            temp = None
        else:
            print("Value not found")

    def search(self, data):
        temp = self.head
        while temp:
            if temp.data == data:
                print(f"Value {data} found")
                return
            temp = temp.next
        print(f"Value {data} not found")
```

```
 def print_list(self):
        temp = self.head
        while temp:
            print(temp.data, end=" <-> ")
            temp = temp.next
        print("None")

def menu():
    dll = DoublyLinkedList()
    while True:
        print("\n1. Insert\n2. Delete\n3. Search\n4. Print List\n5. Exit")
        choice = int(input("Enter choice: "))
        if choice == 1:
            data = int(input("Enter value to insert: "))
            dll.insert(data)
        elif choice == 2:
            data = int(input("Enter value to delete: "))
            dll.delete(data)
        elif choice == 3:
            data = int(input("Enter value to search: "))
            dll.search(data)
        elif choice == 4:
            dll.print_list()
        elif choice == 5:
            break
        else:
            print("Invalid choice")

menu()
```

```
1. Insert
2. Delete
3. Search
4. Print List
5. Exit
Enter choice: 1
Enter value to insert:
10
```

```
1. Insert
2. Delete
3. Search
4. Print List
5. Exit
Enter choice: 1
Enter value to insert:
20
```

```
1. Insert
2. Delete
3. Search
4. Print List
5. Exit
Enter choice: 4
10 <-> 20 <-> None
```

Q.7 Write a program to reverse the given string using list

Program:

```
def reverse_string(s):
    char_list = list(s)
    char_list.reverse()
    return ''.join(char_list)

input_string = input("Enter a string: ")
reversed_string = reverse_string(input_string)
print("Reversed string:", reversed_string)
```

Output:

```
Enter a string: Hi
Reversed string: iH
```

Q.8 Write a program to create singly linear linked list for insert delete search operation

Program:

```
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 delete(self, key):
        temp = self.head
        if temp and temp.data == key:
            self.head = temp.next
            temp = None
            return
        prev = None
        while temp and temp.data != key:
            prev = temp
            temp = temp.next
        if not temp:
            print("Value not found")
            return
        prev.next = temp.next
        temp = None

    def search(self, key):
        temp = self.head
        while temp:
            if temp.data == key:
                print(f"Value {key} found")
                return
            temp = temp.next
        print(f"Value {key} not found")

    def print_list(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")
```

```
def menu():
    sll = SinglyLinkedList()
    while True:
        print("\n1. Insert\n2. Delete\n3. Search\n4. Print List\n5. Exit")
        choice = int(input("Enter choice: "))
        if choice == 1:
            data = int(input("Enter value to insert: "))
            sll.insert(data)
        elif choice == 2:
            data = int(input("Enter value to delete: "))
            sll.delete(data)
        elif choice == 3:
            data = int(input("Enter value to search: "))
            sll.search(data)
        elif choice == 4:
            sll.print_list()
        elif choice == 5:
            break
        else:
            print("Invalid choice")

menu()
```

Output:

```
1. Insert
2. Delete
3. Search
4. Print List
5. Exit
Enter choice: 1
Enter value to insert: 10

1. Insert
2. Delete
3. Search
4. Print List
5. Exit
Enter choice: 1
Enter value to insert: 20

1. Insert
2. Delete
3. Search
4. Print List
5. Exit
Enter choice: 4
20 -> 10 -> None
```

Q.9 Write a program to create queue using linked list and manipulate it

Program:

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

class Queue:
    def __init__(self):
        self.front = None
        self.rear = None

    def enqueue(self, data):
        new_node = Node(data)
        if not self.rear:
            self.front = self.rear = new_node
            return
        self.rear.next = new_node
        self.rear = new_node

    def dequeue(self):
        if not self.front:
            print("Queue is empty")
            return
        temp = self.front
        self.front = self.front.next
        if not self.front:
            self.rear = None
        temp = None

    def display(self):
        if not self.front:
            print("Queue is empty")
            return
        temp = self.front
        while temp:
            print(temp.data, end=" <- ")
            temp = temp.next
        print("None")
```

```
def menu():
    q = Queue()
    while True:
        print("\n1. Enqueue\n2. Dequeue\n3. Display Queue\n4. Exit")
        choice = int(input("Enter choice: "))
        if choice == 1:
            data = int(input("Enter value to enqueue: "))
            q.enqueue(data)
        elif choice == 2:
            q.dequeue()
        elif choice == 3:
            q.display()
        elif choice == 4:
            break
        else:
            print("Invalid choice")

menu()
```

Output:

```
1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 10

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 1
Enter value to enqueue: 20

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 3
10 <- 20 <- None

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 2

1. Enqueue
2. Dequeue
3. Display Queue
4. Exit
Enter choice: 3
    20  - None
```

Q.10 Write a program to reverse stack using list

Program:

```
def reverse_stack(stack):
    if len(stack) > 0:
        temp = stack.pop()
        reverse_stack(stack)
        insert_at_bottom(stack, temp)

def insert_at_bottom(stack, item):
    if len(stack) == 0:
        stack.append(item)
    else:
        temp = stack.pop()
        insert_at_bottom(stack, item)
        stack.append(temp)

stack = [1, 2, 3, 4, 5]
print("Original Stack:", stack)
reverse_stack(stack)
print("Reversed Stack:", stack)
```

Output:
Original Stack: [1, 2, 3, 4, 5]
Reversed Stack: [5, 4, 3, 2, 1]

Q.11 Write a program to create Binary search tree and traverse it using recursive preorder, inorder, postorder methods

Program:

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

def insert(root, data):
    if not root:
        return Node(data)
    if data < root.data:
        root.left = insert(root.left, data)
    else:
        root.right = insert(root.right, data)
    return root

def preorder(root):
    if root:
        print(root.data, end=" ")
        preorder(root.left)
        preorder(root.right)

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

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

root = None
values = [10, 5, 20, 3, 7]
for val in values:
    root = insert(root, val)

print("Preorder Traversal:")

preorder(root)

print("\nInorder Traversal:")
inorder(root)

print("\nPostorder Traversal:")
postorder(root)
```

Output:

```
Preorder Traversal:
10 5 3 7 20
Inorder Traversal:
3 5 7 10 20
Postorder Traversal:
3 7 5 20 10
```

Q.12 Write python program to implement of Queues Using Arrays

Program:

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

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

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

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

    def display(self):
        if self.is_empty():
            print("Queue is empty")
        else:
            print("Queue:", self.queue)


q = Queue()
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)

q.display()

print("Dequeued:", q.dequeue())

q.display()
```

Output:

```
Queue: [10, 20, 30]
Dequeued: 10
Queue: [20, 30]
```

Q.13 Implement Bubble sort to sort a list of numbers.

Program:

```
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]


arr = [64, 25, 12, 22, 11]
bubble_sort(arr)
print("Sorted list:", arr)
```

Output:
Sorted list: [11, 12, 22, 25, 64]

Q.14 Write a program to implement binary search using array

Program:

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

# Example usage:
arr = [1, 3, 5, 7, 9, 11, 13, 15]
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")
```

Output:

```
Element 7 found at index 3
```

Q.15 Write a program to implement adjacency matrix and display using linked list.

Program:

```
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.adj_matrix = [[0] * self.V for _ in range(self.V)]

    def add_edge(self, u, v):
        self.adj_matrix[u][v] = 1
        self.adj_matrix[v][u] = 1  # For undirected graph

    def display(self):
        for row in self.adj_matrix:
            print(" ".join(map(str, row)))

# Example usage:
g = Graph(5)
g.add_edge(0, 1)
g.add_edge(0, 4)
g.add_edge(1, 2)
g.add_edge(1, 3)
g.add_edge(2, 4)

g.display()
```

Output:

```
0 1 0 0 1
1 0 1 1 0
0 1 0 0 1
0 1 0 0 0
1 0 1 0 0
```