Experiment No:5

Aim: To implement menu driven programs for Link List, Stack and Queue in python

Theory:

A linked list is a sequential collection of data elements, which are connected together via links. A linked list consists of independent nodes containing any type of data and each node holds a reference or a link to the next node in the list.

The beginning node of a linked list is called the **head** and the end node is called the **tail.** All nodes of a linked list are independent and are not stored contagiously in memory.

Types of Linked Lists
There are 4 types of linked lists that can be created in python.
Singly Linked List
Circular Singly Linked List
Doubly Linked List
Circular Doubly Linked List

Stack:
In python, the stack is an abstract data structure that stores elements linearly. The items in a stack follow the Last-In/First-Out (LIFO) order. This means that the last element to be inserted in a stack will be the first one to be removed.

Stack Operations
Various operations can be performed on a stack in python.
Create Stack
Push

Pop
Peek
isEmpty
isFull
deleteStack

Queue

In python, the queue is an abstract data structure that stores elements linearly. The items in a queue follow the First-In/First-Out (FIFO) order. This means that the first element to be inserted in a queue will be the first one to be removed.

Queue Operations

Various operations can be performed on a queue in python.
Create Queue .
Enqueue
Dequeue
Peek
isEmpty
isFull
deleteQueue

PROGRAM

Program 5.1: Stack

```
# Program introduction statement
class Student:
    def __init__(self, name, roll_number, grade):
        self.name = name
        self.roll_number = roll_number
        self.grade = grade

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

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

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()
        else:
            return None

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

    def display_details(self):
        if not self.is_empty():
            print("Student Details:")
            for student in reversed(self.stack):
                print(f"Name: {student.name}, Roll Number: {student.roll_number}, Grade:
{student.grade}")
        else:
            print("Stack is empty.")

def main():
    student_stack = StudentStack()

    while True:
        print("\n1. Insert Student Details")
        print("2. Delete Student Details")
        print("3. Display All Student Details")
        print("4. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
```

```
        name = input("Enter student name: ")
        roll_number = input("Enter roll number: ")
        grade = input("Enter grade: ")
        student = Student(name, roll_number, grade)
        student_stack.push(student)
        print("Student details inserted successfully.")
    elif choice == 2:
        deleted_student = student_stack.pop()
        if deleted_student:
            print("Deleted Student Details:")
            print(f"Name: {deleted_student.name}, Roll Number: {deleted_student.roll_number}, Grade:
{deleted_student.grade}")
        else:
            print("Stack is empty. No student details to delete.")
    elif choice == 3:
        student_stack.display_details()
    elif choice == 4:
        print("Exiting program...")
        break
    else:
        print("Invalid choice. Please enter a valid option.")

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

OUTPUT:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\stack.py"

1. Insert Student Details
2. Delete Student Details
3. Display All Student Details
4. Exit
Enter your choice: 1
Enter student name: yash Kerkar
Enter roll number: 67
Enter grade: A
Student details inserted successfully.

1. Insert Student Details
2. Delete Student Details
3. Display All Student Details
4. Exit
Enter your choice: 3
Student Details:
Name: yash Kerkar, Roll Number: 67, Grade: A

1. Insert Student Details
2. Delete Student Details
3. Display All Student Details
4. Exit
Enter your choice:
```

Program 5.2:Queue

```
# Import Python Package
from queue import Queue

# Program introduction statement
print("Simple QUEUE Data Structure Program")

# Initial empty QUEUE
queue = Queue()

# Display Menu with Choices
while True:
    print("\nSELECT APPROPRIATE CHOICE")
    print("1. PUT Element into the Queue")
    print("2. GET Element from the Queue")
    print("3. Display Elements of the Queue")
    print("4. Exit")

    # Taking input from the user regarding choice
    choice = int(input("Enter the Choice:"))

    # USER enter option 1 then PUT elements into the QUEUE
    if choice == 1:
        # put() function to PUT elements into the QUEUE
        queue.put("Monday")       # PUT element Monday
        queue.put("Tuesday")      # PUT element Tuesday
        queue.put("Wednesday")    # PUT element Wednesday
        queue.put("Thursday")     # PUT element Thursday
        queue.put("Friday")       # PUT element Friday
        queue.put("Saturday")     # PUT element Saturday
        queue.put("Sunday")       # PUT element Sunday
        queue.put('8')            # PUT element 8
        print('\nTotal 8 elements PUT into the QUEUE')

    # USER enter option 2 then GET one element from the QUEUE
    elif choice == 2:
        if queue.empty():
            # Check whether QUEUE is Empty or not
            print('The QUEUE is EMPTY No element to GET out')
        else:
            # get() function to GET element out from the QUEUE in FIFO order
            print('\nElement GET out from the QUEUE is:')
            print(queue.get())   # Display the element which is GET out from the QUEUE
```

```
# USER enter option 3 then display the QUEUE
elif choice == 3:
    if queue.empty():
        # Check whether QUEUE is Empty or not
        print('The QUEUE is initially EMPTY') # Display this message if QUEUE is Empty
    else:
        print("The Size of the QUEUE is: ", queue.qsize()) # Compute the size of the QUEUE
        print('\nQUEUE elements are as follows:')
        print(list(queue.queue))    # Display all the QUEUE elements

# User enter option 4 then EXIT from the program
elif choice == 4:
    break

# Shows ERROR message if the choice is not in between 1 to 4
else:
    print("Oops! Incorrect Choice")
```

OUTPUT:

```
Total 8 elements PUT into the QUEUE

SELECT APPROPRIATE CHOICE
1. PUT Element into the Queue
2. GET Element from the Queue
3. Display Elements of the Queue
4. Exit
Enter the Choice:2

Element GET out from the QUEUE is:
Monday

SELECT APPROPRIATE CHOICE
1. PUT Element into the Queue
2. GET Element from the Queue
3. Display Elements of the Queue
4. Exit
Enter the Choice:2

Element GET out from the QUEUE is:
Tuesday

SELECT APPROPRIATE CHOICE
1. PUT Element into the Queue
2. GET Element from the Queue
3. Display Elements of the Queue
4. Exit
Enter the Choice:3
The Size of the QUEUE is:  6
```

Program 5.3:Linked List

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

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

    def append(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
```

```
last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node

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

    def delete(self, data):
        if not self.head:
            return


        if self.head.data == data:
            self.head = self.head.next
            return
        current_node = self.head
        while current_node.next:
            if current_node.next.data == data:
                current_node.next = current_node.next.next
                return
            current_node = current_node.next

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


# Example usage:
if __name__ == "__main__":
    linked_list = LinkedList()
    linked_list.append(1)
    linked_list.append(2)
    linked_list.append(3)
    linked_list.prepend(0)
    linked_list.print_list()
    linked_list.delete(2)
    linked_list.print_list()
```

OUTPUT:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\linkedlist.py"
0 -> 1 -> 2 -> 3 -> None
0 -> 1 -> 3 -> None
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

Conclusion:

The implementation of menu-driven programs for linked lists, stacks, and queues inPython has demonstrated their versatility and efficiency in managing data structures. Through this experiment, we have gained insights into the practical applications of these fundamental data structures, paving the way for further exploration and optimization in programming solutions.