|  |  |
| --- | --- |
|  | **Manav Rachna University** |
| **Lab Assignment 5** |
| **Subject:** Analysis and Design of Algorithms **Subject Code**:  **Semester: VI** | |

**Learning Objective:** Students would be Able to use stacks, queues in problem-solving (e.g., the evaluation of postfix expressions)

**Learning Outcome:** To implement the data structures for representing a graph.

1. Write a program to implement the following operations on stack through Linked List:
   1. Push
   2. POP

Code:

class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.next = None

class Stack:

    def \_\_init\_\_(self):

        self.top = None

    def push(self, data):

        new\_node = Node(data)

        new\_node.next = self.top

        self.top = new\_node

        print(f"Pushed {data} onto stack")

    def pop(self):

        if self.top is None:

            print("Stack Underflow! Cannot pop from empty stack.")

            return None

        popped\_data = self.top.data

        self.top = self.top.next

        print(f"Popped {popped\_data} from stack")

        return popped\_data

    def display(self):

        if self.top is None:

            print("Stack is empty")

            return

        temp = self.top

        print("Stack elements:")

        while temp:

            print(temp.data, end=" -> ")

            temp = temp.next

        print("None")

stack = Stack()

stack.push(10)

stack.push(20)

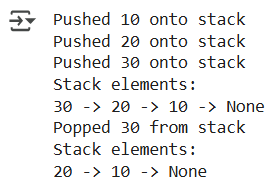
stack.push(30)

stack.display()

stack.pop()

stack.display()

Output:



1. Write a program to implement the following operations on Queue using Linked list:
   1. Insertion at rear
   2. Deletion from front

Code:

class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.next = None

class Queue:

    def \_\_init\_\_(self):

        self.front = self.rear = None

    def enqueue(self, data):

        new\_node = Node(data)

        if self.rear is None:

            self.front = self.rear = new\_node

        else:

            self.rear.next = new\_node

            self.rear = new\_node

        print(f"Inserted {data} at rear")

    def dequeue(self):

        if self.front is None:

            print("Queue Underflow! Cannot dequeue from empty queue.")

            return None

        dequeued\_data = self.front.data

        self.front = self.front.next

        if self.front is None:

            self.rear = None

        print(f"Deleted {dequeued\_data} from front")

        return dequeued\_data

    def display(self):

        if self.front is None:

            print("Queue is empty")

            return

        temp = self.front

        print("Queue elements:")

        while temp:

            print(temp.data, end=" -> ")

            temp = temp.next

        print("None")

queue = Queue()

queue.enqueue(10)

queue.enqueue(20)

queue.enqueue(30)

queue.display()

queue.dequeue()

queue.display()

Output:

