KALYANI GOVERNMENT ENGINEERING COLLEGE

Department of Computer Application

Lab Assignment 6

Topic - Linked List

Name: Madhusudan Chand
Roll No.: 10271023013
Registration No.: 231020510013
Stream: MCA
Semester: 2^{nd}

Paper: Date Structure Lab with python
Paper Code: MCAN-291

Contents

6.1 Write a Python program that implements stack as a linked list.

Ans -

Code

```
import sys
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
    def __del__(self):
        print("Destruction")


class Stack(Node):
    def __init__(self):
        self.head = None

    def push(self,x):
        nwnode = Node(x)
        if self.head == None:
            self.head = nwnode
            return
        temp = self.head
        self.head = nwnode
        self.head.next = temp

    def pop(self):
        if self.head == None:
            print("Underflow")
            return
        temp = self.head
        self.head = temp.next
        print(temp.data ,"deleted.")
        del temp
```

```
    def display(self):
        print("The stack -> ")
        if self.head == None:
            print("Empty stack")
            return
        temp = self.head
        while temp:
            print(temp.data , end = " ")
            temp = temp.next

        print()

if __name__ == "__main__":
    s=Stack()
    print("STACK")
    while True:
        print("1.Push")
        print("2.Pop")
        print("3.Display")
        print("4.Exit")
        print("Enter your choice")
        ch = int(input())

        if ch == 1:
            print("Enter value")
            s.push(int(input()))
        elif ch == 2:
            s.pop()
        elif ch == 3:
            s.display()
        elif ch == 4:
            sys.exit()
        else:
            print("Invalid")
```

Output

```
STACK
1.Push
2.Pop
3.Display
4.Exit
Enter your choice
1
Enter value
53
1.Push
2.Pop
3.Display
4.Exit
Enter your choice
1
Enter value
90
1.Push
2.Pop
3.Display
4.Exit
Enter your choice
1
Enter value
52
1.Push
2.Pop
3.Display
4.Exit
Enter your choice
3
The stack ->
52 90 53
1.Push
2.Pop
```

```
3.Display
4.Exit
Enter your choice
2
52 deleted.
Destruction
1.Push
2.Pop
3.Display
4.Exit
Enter your choice
2
90 deleted.
Destruction
1.Push
2.Pop
3.Display
4.Exit
Enter your choice
4
Destruction
Destruction
```

6.2 Write a Python program that implements queue as a linked list.

Ans -

Code

```
import sys
class Node:
    def __init__(self,data):
        self.data = data
        self.next=None
    def __del__(self):
        print("Destructor")
```

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

    def isempty(self):
        return self.rear ==None

    def insertInQ(self,data):
        nwnode = Node(data)
        if self.rear == None:
            self.rear = self.front = nwnode
            return
        self.rear.next = nwnode
        self.rear = nwnode

    def deleteFromQ(self):
        if self.front == None:
            print("Queue is empty")
            return

        temp=self.front
        self.front = temp.next
        print("Delted item",temp.data)
        del temp

    def displayQ(self):
        if self.front==None and self.rear == None:
            print("Empty queue")
            return
        temp = self.front
        print("The queue - >")
        while temp:
            print(temp.data)
            temp = temp.next
```

```
if __name__ == "__main__":
    q = Queue()
    print("QUEUE")
    while True:
        print("1.Insert")
        print("2.Delete")
        print("3.Display")
        print("4.Exit")
        print("Enter your choice")
        ch = int(input())

        if ch == 1:
            q.insertInQ(int(input("Enter value : \n")))
        elif ch == 2:
            q.deleteFromQ()
        elif ch == 3:
            q.displayQ()
        elif ch == 4:
            sys.exit()
        else:
            print("Invalid")
```

Output

```
QUEUE
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
1
Enter value :
82
1.Insert
2.Delete
3.Display
```

```
4.Exit
Enter your choice
1
Enter value :
21
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
3
The queue - >
82
21
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
2
Delted item 82
Destructor
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
3
The queue - >
21
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
```

```
4
Destructor
Destructor
```

6.3 Write a Python program that implements circular queue as a linked list..

Ans -

Code

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

class CircularQueue:
    def __init__(self):
        self.start = None

    def insert_cc_queue(self):
        item = int(input("Enter the value to be inserted: "))
        ptr = Node(item)
        if self.start is None:
            ptr.next = ptr
            self.start = ptr
        else:
            temp = self.start
            while temp.next != self.start:
                temp = temp.next
            ptr.next = temp.next
            temp.next = ptr
    def delete_cc_queue(self):
        if self.start is None:
            print("Underflow")
            return
```

```
ptr = self.start
        if self.start.next == self.start:
            self.start = None
        else:
            temp = self.start
            while temp.next != self.start:
                temp = temp.next
            self.start = self.start.next
            temp.next = self.start
        print("Deleted element is:", ptr.data)
        del ptr
    def display(self):
        if self.start is None:
            print("Queue is empty.")
            return

        ptr = self.start
        while True:
            print("Data is", ptr.data)
            ptr = ptr.next
            if ptr == self.start:
                break
if __name__ == "__main__":
    cq = CircularQueue()
    while True:
        print("\n1.Insert in the circular queue.")
        print("2.Delete from the circular queue.")
        print("3.Display the queue.")
        print("4.Exit.")
        choice = int(input("Enter your choice: "))
        if choice == 1:
            cq.insert_cc_queue()
        elif choice == 2:
            cq.delete_cc_queue()
        elif choice == 3:
```

```
        cq.display()
            elif choice == 4:
                break
            else:
                print("Invalid choice.")
```

Output

```
1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 1
Enter the value to be inserted: 62

1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 1
Enter the value to be inserted: 78

1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 1
Enter the value to be inserted: 62

1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 1
Enter the value to be inserted: 89
```

```
1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 3
Data is 62
Data is 78
Data is 62
Data is 89

1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 2
Deleted element is: 62

1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 3
Data is 78
Data is 62
Data is 89

1.Insert in the circular queue.
2.Delete from the circular queue.
3.Display the queue.
4.Exit.
Enter your choice: 4
```

6.4 Write a Python program that implements deque using a linked list..

Ans -

Code

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

class Dqueue:
    def __init__(self):
        self.rear = None
        self.front = None
        self.size = 0
    def isEmpty(self):
        return self.front == None
    def insertFrontend(self,data):
        nwnode = Node(data)
        if nwnode is None:
            print("Overflow")
            return
        if self.front == None:
            self.rear = self.front = nwnode
        else:
            nwnode.next = self.front
            self.front.prev=nwnode
            self.front=nwnode
```

```
def insertRearend(self,data):
        nwnode = Node(data)
        if nwnode is None:
            print("Overflow")
            return
        if self.rear == None:
            self.rear = self.front = nwnode
        else:
            nwnode.prev = self.rear
            self.rear.next=nwnode
            self.rear=nwnode
        pass

    def deleteFrontEnd(self):
        if self.front == None:
            print("Underflow")
            return

        temp = self.front
        self.front = self.front.next

        if self.front == None:
            self.rear = None
            print(temp.data ,"removed")
            del temp
            return
        self.front.prev = None
        print(temp.data ,"removed")
        del temp
    def deleteRearEnd(self):
        if self.rear == None:
            print("Underflow")
            return
        temp = self.rear
        self.rear = self.rear.prev
```

```
        if self.rear == None:
            self.front = None
            print(temp.data ,"removed")
            del temp
            return
        self.rear.next = None
        print(temp.data ,"removed")
        del temp
        pass

    def getFront(self):
        if self.isEmpty():
            print("empty queue")
            return
        print("Front value",self.front.data)

    def getRear(self):
        if self.isEmpty():
            print("empty queue")
            return
        print("Rear value",self.rear.data)

    def display(self):
        if self.isEmpty():
            print("Empty queue")
            return
        temp = self.front
        while temp:
            print(temp.data,end=" ")
            temp = temp.next

        print()
```

```
if __name__=="__main__":
    q = Dqueue()
    while True:
        print("1.Insert into Front")
        print("2.Insert into Rear")
        print("3.Delete from Front")
        print("4.Delete from rear")
        print("5.Get Front")
        print("6.Get Rear")
        print("7.Display")
        print("8.Exit")

        print("Enter your choice")
        ch = int(input())

        if ch==1:
            q.insertFrontend(int(input("Enter value\n")))
        elif ch==2:
            q.insertRearend(int(input("Enter value\n")))
        elif ch==3:
            q.deleteFrontEnd()
        elif ch==4:
            q.deleteRearEnd()
        elif ch==5:
            q.getFront()
        elif ch==6:
            q.getRear()
        elif ch==7:
            q.display()
        elif ch==8:
            break
        else:
            print("Invalid choice")
```

Output

```
1.Insert into Front
2.Insert into Rear
3.Delete from Front
4.Delete from rear
5.Get Front
6.Get Rear
7.Display
8.Exit
Enter your choice
1
Enter value
62
Enter your choice
2
Enter value
90
...
Enter your choice
1
Enter value
72
....
Enter your choice
1
Enter value
89
....
Enter your choice
2
Enter value
953
...
Enter your choice
7
89 72 62 90 953
```

6.5 Write a Python program that implements a priority queue using a linked list.

Ans -

Code

```
class Node:
    def __init__(self,val,priority):
        self.val = val
        self.priority = priority

    def __del__(self):
        print("Destructor")

    def __repr__(self) -> str:
        return "Node(val: {}, pri: {})".
                                format(self.val,self.priority)


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

    def priorityCheck(self,p1,p2):
        return p1<p2
    def insert(self,val,priority):
        data = Node(val,priority)
        index_pro=0
        while index_pro < len(self.storage):
            if self.priorityCheck
                (self.storage[index_pro].priority,priority):
                break
            index_pro+=1
        self.storage.insert(index_pro,data)
```

```
    def peek(self):
            return self.storage[-1].val

    def pop(self):
        return self.storage.pop().val

    def display(self):
        print("Priority Queue")
        for item in self.storage:
            print("Value:",item.val," priority:",item.priority)

if __name__=="__main__":
    pq=PriorityQueue()
    pq.insert(78,3)
    pq.insert(873,1)
    pq.insert(781,2)
    print(pq.display())

    print("Remove item from queue",pq.pop())
    print("peek value",pq.peek())
    print(pq.display())
```

Output

```
Priority Queue
Value: 78  priority: 3
Value: 781  priority: 2
Value: 873  priority: 1
Destructor
Remove item from queue 873
peek value 781
Priority Queue
Value: 78  priority: 3
Value: 781  priority: 2
Destructor
Destructor
```