MEGHNAD SAHA INSTITUTE OF TECHNOLOGY

*Techno Complex, Madurdaha,Beside NRI Complex, Post-Uchhepota, Kolkata 700 150*

LABORATORY NOTE BOOK

MAKAUT EVEN SEMESTER 2025



[MASTERS OF COMPUTER APPLICATION]

[DATA STRUCTURE LAB USING PYTHON (MCAN-291)]

[RUPAK SARKAR]

ROLL NO: 14271024036 REGN. NO.: 241420510045

STREAM: MCA SEMESTER: II (2ND)

YEAR: 1ST YearSESSION: 2024-2026



MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY



MEGHNAD SAHA INSTITUTE OF TECHNOLOGY

*Techno Complex,. Madurdaha,Beside NRI Complex, Post-Uchhepota, Kolkata 700 150*

“LIST OF ASSIGNMENT/EXPERIMENT SUBMISSION DETAILS”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL.**  **NO.** | **ASSIGNMENT / EXPERIMENT NAME** | **DATE OF EXPERIMENT** | **DATE OF SUBMISION** | **CHECKED BY** | **REMARKS (ANY DEVIATION REGARDING SUBMISSION DATES, CONTENT, FORMAT, ETC)** |
| 1. | WAP to implement Stack. | 21/02/2025 | 28/02/2025 |  |  |
| 2. | WAP to implement Linear Queue. | 21/02/2025 | 28/02/2025 |  |  |
| 3. | WAP to implement Circular Queue. | 21/02/2025 | 28/02/2025 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |
| --- |
| OBSERVATIONS / COMMENTS ON THE OVERALL PERFORMANCE: |

Signature in full with date Signature in full with date

**Faculty / Technical Assistant Lab Examiner**

**Q.1. Write a python program to implement Stack.**

Ans:

class Stack:

def \_\_init\_\_(self, MAX):

self.size=MAX

self.arr=[None]\*MAX

self.top=-1

def push(self, item):

if (self.top== self.size-1):

print("\nStack is Full!")

else:

self.top=self.top+1

self.arr[self.top]=item

def pop(self):

if (self.top==-1):

print("\nStack is Empty!")

else:

x=self.arr[self.top]

self.top=self.top-1

return x

def display(self):

if (self.top==-1):

print("\nNothing to display!")

else:

print("\nElements in the Stack are:")

for i in range(0,self.top+1):

print(self.arr[i], end=" ")

if \_\_name\_\_=="\_\_main\_\_":

st=Stack(7)

st.display()

st.push(2)

st.push(4)

st.push(6)

st.display()

print("\nFirst deleted element is:",st.pop())

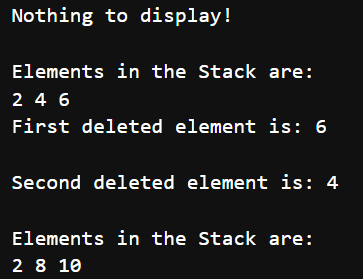
print("\nSecond deleted element is:",st.pop())

st.push(8)

st.push(10)

st.display()

Output:



**Q.2. Write a python program to implement Linear Queue.**

Ans:

class Node:

def \_\_init\_\_(self,data):

self.info=data

self.link=None

class Single\_Linked\_List:

def \_\_init\_\_(self):

self.head=None

self.tmp=None

def create(self,item):

new\_node=Node(item)

if self.head is None:

self.head=new\_node

self.tmp=self.head

else:

self.tmp.link=new\_node

self.tmp=self.tmp.link

def insert\_at\_beg(self,item):

new\_node=Node(item)

if self.head is None:

self.head=new\_node

return

else:

new\_node.link=self.head

self.head=new\_node

def insert\_at\_end(self, item):

new\_node=Node(item)

tmp=self.head

while tmp.link:

tmp=tmp.link

tmp.link=new\_node

def insert\_at\_pos(self,item,pos):

new\_node=Node(item)

if (pos<1):

print("\nInsert a valid position")

return

if (pos==1):

new\_node.link=self.head

self.head=new\_node

else:

tmp=self.head

for i in range(1,pos-1):

tmp=tmp.link

new\_node.link=tmp.link

tmp.link=new\_node

def insert\_after(self, item, value):

new\_node=Node(item)

tmp=self.head

while (tmp.info!=value):

tmp=tmp.link

new\_node.link=tmp.link

tmp.link=new\_node

def insert\_before(self, item, value):

new\_node=Node(item)

tmp=self.head

while (tmp.info!=value):

tmp2=tmp

tmp=tmp.link

new\_node.link=tmp2.link

tmp2.link=new\_node

def display(self):

pt=self.head

print("\nElements in the Linked List are:")

while pt is not None:

print(pt.info,end=" ")

pt=pt.link

if \_\_name\_\_=="\_\_main\_\_":

sl=Single\_Linked\_List()

n=int(input("\nHow many number you want to insert in the Linked List:"))

for i in range(n):

x=int(input("\nEnter the value of node %d:"%(i+1)))

sl.create(x)

sl.display()

sl.insert\_at\_beg(25)

sl.display()

sl.insert\_at\_end(35)

sl.display()

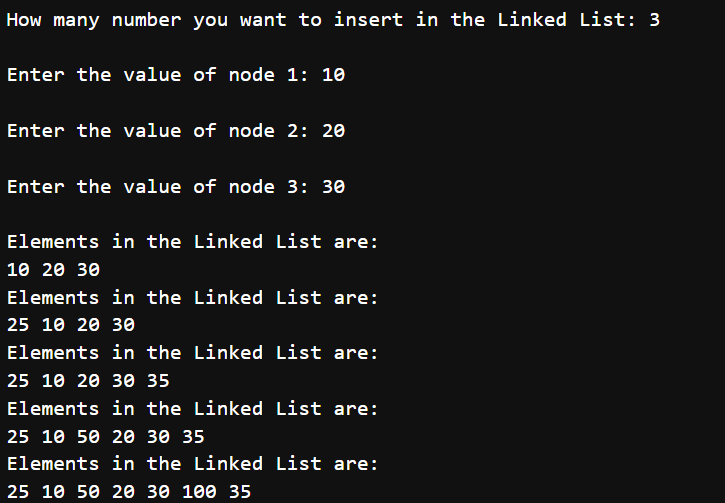
sl.insert\_at\_pos(50,3)

sl.display()

sl.insert\_after(100,30)

sl.display()

Output:



**Q.3. Write a python program to implement Circular Queue.**

Ans:

class Circular\_Queue:

def \_\_init\_\_(self, Max):

self.size=Max

self.queue=[None]\*Max

self.front=self.rear=-1

def enqueue(self,item):

if(self.front==(self.rear+1)% self.size):

print("\nCircular Queue is Full!")

elif(self.front==-1):

self.front=self.rear=0

self.queue[self.rear]=item

else:

self.rear=(self.rear+1)%self.size

self.queue[self.rear]=item

def dequeue(self):

if(self.front==-1):

print("\nCircular Queue is Empty.")

elif (self.front==self.rear):

x=self.queue[self.front]

self.front=self.rear=-1

return x

else:

x=self.queue[self.front]

self.front=(self.front+1) % self.size

return x

def display(self):

if(self.front==-1):

print("\nCircular Queue is Empty!")

elif(self.rear>=self.front):

print("\nElements in the Circular Queue are: ")

for i in range(self.front,self.rear+1):

print(self.queue[i], end=" ")

else:

print("\n Elements in the Cicular Queue are: ")

for i in range(self.front,self.size):

print(self.queue[i], end=" ")

for i in range(0, self.rear+1):

print(self.queue[i], end=" ")

if \_\_name\_\_=="\_\_main\_\_":

cq=Circular\_Queue(6)

cq.display()

cq.enqueue(2)

cq.enqueue(4)

cq.enqueue(6)

cq.enqueue(4.5)

cq.display()

print("\nFirst Deleted value=", cq.dequeue())

print("\nSecond Deleted value=", cq.dequeue())

cq.display()

cq.enqueue(12)

cq.enqueue(3.9)

cq.enqueue(10)

cq.enqueue(20)

cq.display()

cq.enqueue(8)

Output:

