**SSN College of Engineering**

**Department of Information Technology**

**UIT2201 — Programming and Data Structures 2022 – 2023**

**Exercise — 10**

**Part A**

**1)Design and implement Linked List with the following operations.**

**• Isempty**

**• Display**

**• Find**

**• append**

**• Insert by pos**

**• Delete by pos**

**• Insert by specifying previous value.**

**• Delete by specifying previous value.**

**Code:**

class Node:

    '''This class is used for creating the Node'''

    \_\_slots\_\_=["item","next"]

    def \_\_init\_\_(self,item=None,next=None):     #Initializing the Constructor

        self.item=item

        self.next=next

class SingleList:

    '''This class is used for creating the  linked list'''

    def \_\_init\_\_(self):                          #Initializing the Constructor

        self.head=self.tail=Node()

    def isempty(self):                           #Check wheather the linked list is empty

        return self.head==self.tail

    def append(self,ele):                        #For Adding the element in the end to the linked list

        temp=Node(ele)

        self.tail.next=temp

        self.tail=temp

    def display(self):                           #For Displaying the linked list

        pos=self.head.next

        while (pos is not None):

            print(pos.item,end=",")

            pos=pos.next

    def find(self,ele):                          #To Find the element from linked list

        pos=self.head.next

        while (pos is not None):

            if (pos.item==ele):

                return pos

            pos=pos.next

        return -1

    def insert(self,ele,position):               #To Insert the element in any position of linked list

        pos=self.head.next

        for i in range(position-1):

            pos=pos.next

        temp=Node(ele)

        temp.next=pos.next

        pos.next=temp

    def prev\_ele\_find(self,data):                #It is used to find the previous element's position

        pos=self.head.next

        while (pos is not None):

            if (pos.next.item==data):

                return pos

            pos=pos.next

        return -1

    def delete(self,data):                        #To Delete the element from the link list

        prev\_ele=self.prev\_ele\_find(data)

        del\_ele=prev\_ele.next

        prev\_ele.next=del\_ele.next

#Driver Code

s=SingleList()                                    #Creating the Object to Class SingleList

print("Checking Empty Condition At Beginning")

print(s.isempty())

print("Adding the Elements")

s.append(1)

s.append(2)

s.append(3)

print(s.display())

print("Checking Empty Condition")

print(s.isempty())

print("Finding the Element 2")

print(s.find(2))

print("Inserting the Element 10 at index 2")

s.insert(10,2)

print(s.display())

print("Inserting the Element 12 at index 1")

s.insert(12,1)

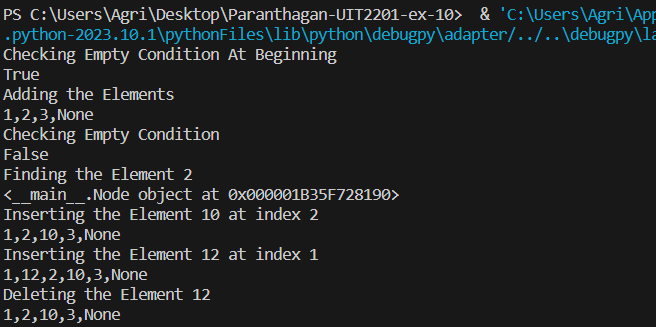
print(s.display())

print("Deleting the Element 12")

s.delete(12)

print(s.display())

**Output:**



**2)Design and implement Linked stack and Queue ADT with various operations.**

**Code:**

class Node:

    '''This class is used for creating the Node'''

    \_\_slots\_\_=["item","next"]

    def \_\_init\_\_(self,item=None,next=None):         #Initializing the Constructor

        self.item=item

        self.next=next

class LinkStack:

    '''This class is used for creating the  linked stack'''

    def \_\_init\_\_(self):                             #Initializing the Constructor

        self.top=Node()

        self.size=0

    def isempty(self):                              #Check wheather the linked stack is empty

        return self.top.next is None

    def peek(self):                                 #It gives the peek value of linked stack

        return self.top.item

    def push(self,ele):                             #For Adding the element in the end to the linked stack

        temp=Node(ele)

        temp.next=self.top.next

        self.top.next=temp

        self.size+=1

    def pop(self):                                  #For Deleting the element in the end to the linked stack

        if self.isempty():

            return IndexError("Index Out of Range")

        else:

            del\_node=self.top.next

            self.top.next=del\_node.next

            self.size-=1

    def display(self):                              #To Display the linked stack

        pos=self.top.next

        string=""

        while (pos is not None):

            string=string+str(pos.item)

            pos=pos.next

            string=string+","

        return "["+string+"]"

#Driver Code

s=LinkStack()                                       #Creating the object to Class LinkStack

print(s.isempty())

s.push(10)

s.push(12)

s.push(14)

s.push(16)

s.push(18)

print(s.display())

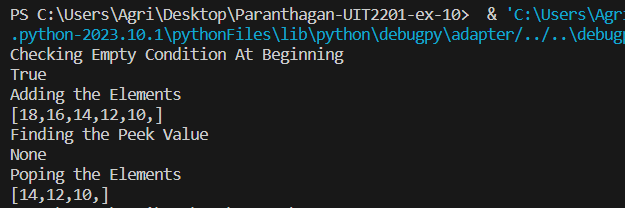
print(s.peek())

s.pop()

s.pop()

print(s.display())

**Output:**

****

**Code:**

class Node:

    '''This class is used for creating the Node'''

    \_\_slots\_\_=["item","next"]

    def \_\_init\_\_(self,item=None,next=None):         #Initializing the Constructor

        self.item=item

        self.next=next

class LinkQueue:

    '''This class is used for creating the  linked queue'''

    def \_\_init\_\_(self):                             #Initializing the Constructor

        self.front=Node()

        self.rear=Node()

        self.size=0

    def isempty(self):                              #Check wheather the linked queue is empty

        return self.size==0

    def enqueue(self,ele):                          #For Adding the element in the end to the linked queue

        temp=Node(ele)

        if self.isempty():

            self.front = temp

            self.rear = temp

        else:

            self.rear.next=temp

            self.rear=temp

        self.size+=1

    def dequeue(self):                               #To Delete the element from the link queue

        if self.isempty():

            return IndexError("Index Out of Range")

        else:

            del\_node=self.front.next

            self.front.next=del\_node.next

        self.size-=1

    def display(self):                               #For Displaying the linked queue

        pos=self.front

        string=""

        while (pos is not None):

            string=string+str(pos.item)

            pos=pos.next

            string=string+","

        return "["+string+"]"

#Driver Code

q=LinkQueue()                                        #Creating the Object to Class LinkQueue

print("Checking Empty Condition At Beginning")

print(q.isempty())

print("Adding the Elements")

q.enqueue(1)

q.enqueue(2)

q.enqueue(4)

q.enqueue(6)

q.enqueue(8)

print(q.display())

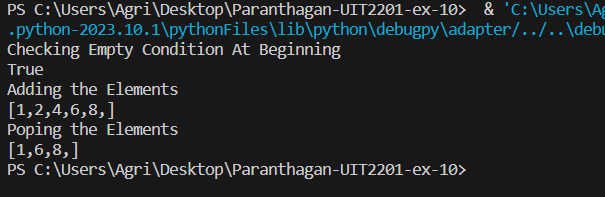
print("Poping the Elements")

q.dequeue()

q.dequeue()

print(q.display())

**Output:**

****