University Of Baltistan Skardu

Submitted By: Basit Ali

Submitted To: Mam Noreen

Date: 08/04/24

Registration No: S23BSCS012

Assignment No: 09 to 11

Department Of Computer Science

Assignment 09

Program 01:

```
# Q1: Implement the Queue using list?
class Queue:
    def __init__(self):
        self.data=[]
        self.counter=0
    def isEmpty(self):
        return  len(self.data)==0
    def enqueue(self,data):
        self.data.insert(0,data)
        self.counter+=1
    def dequeue(self):
        if(self.isEmpty()):
            print("Queue is Empty")
        else:
            self.counter-=1
            return   self.data.pop()
    def getFront(self):
        if(self.isEmpty()):
            print("Queue is Empty")
        else:
            return  self.data[len(self.data)-1]
```

```
    def getRear(self):
        return self.data[0]
    def size(self):
        return self.counter
    def display(self):
        for i in reversed(range(len(self.data))):
            print(self.data[i],end=" ")
        print()
newObj= Queue()
newObj.enqueue(1)
newObj.enqueue(2)
newObj.enqueue(3)
newObj.enqueue(4)
print("\nBefore deleting an element : ")
newObj.display()
print("Rear item is: ",newObj.getRear())
print("Front item is: ",newObj.getFront())
print("Number of item is: ",newObj.size())
newObj.dequeue()
print("After deleting an element, remaining items are : ")
newObj.display()
```

Output Of Program Is:

```
Before deleting an element :
1 2 3 4
Rear item is:  4
Front item is:  1
Number of item is:  4
After deleting an element, remaining items are :
2 3 4
```

Program 02:

```
# Q2: Implement the Queue using Array?
from array import *
class Arrays:
    def __init__(self):
        self.arr=array('i',[]);
    def isEmpty(self):
        return len(self.arr)==0
    def enqueue(self,item):
        self.arr.insert(0,item);
    def dequeue(self):
        if(self.isEmpty()):
            print("Array is Empty")
        else:
            return self.arr.pop();
    def getFront(self):
        return self.arr[len(self.arr)-1]
    def getRear(self):
        return self.arr[0]
    def show(self):
        if self.isEmpty():
            print('Array is Empty')
        else:
            for item in self.arr:
                print(item,end=" ");
Arr=Arrays()
Arr.enqueue(54)
Arr.enqueue(32)
Arr.enqueue(11)
Arr.enqueue(12)
Arr.enqueue(14)
Arr.show()
Arr.dequeue()
print('\nFront Element of Array is : ',Arr.getFront())
print('Rear Element of Array is : ',Arr.getRear())
print('Pop the front element :',Arr.dequeue())
print('Front element :',Arr.getFront())
Arr.show()
```

Output Of Program Is:

```
Front Element of Array is :  32
Rear Element of Array is :  14
Pop the front element : 32
Front element : 11
14 12 11
```

Assignment 10:

Program 01:

```python
# Q1 Implement the Dequeue using Array?
from array import *
class Dequeue:
    def __init__(self):
        self.arr=array('i',[])
    def isEmpty(self):
        return  len(self.arr)==0
    def addFirst(self,element):
        self.arr.insert(0,element)
    def removeFirst(self):
        if not self.isEmpty():
            return self.arr.pop(0)
    def addLast(self,element):
        self.arr.append(element)
    def removeLast(self):
        if not self.isEmpty():
            return self.arr.pop(-1)
    def getFront(self):
        if not self.isEmpty():
            return  self.arr[0]
        else:
            return "linked list is Empty"
```

```
    def getRear(self):
        if not self.isEmpty():
            return  self.arr[-1]
        else:
            return "linked list is Empty"
    def getSize(self):
        if self.isEmpty():
            return "linked list is Empty"
        else:
            return len(self.arr)
    def display(self):
        for i in range(len(self.arr)):
            print(self.arr[i],end=" ")
# Testing the class
newObj=Dequeue()
newObj.addFirst(10)
newObj.addFirst(11)
newObj.addFirst(12)
newObj.addLast(20)
newObj.addLast(21)
newObj.addLast(22)
```

```
print("\nDisplaying the Dequeue Before removeing")
newObj.display()
print("\nSize of list before Deleting",newObj.getSize())
print("Getting First and Rear Before Deleting")
print("First Element :",newObj.getFront())
print("Rear Element :",newObj.getRear())
print("Displaying the Dequeue After removeing 1st and last")
newObj.removeFirst()
newObj.removeLast()
newObj.display()
print("\nSize of list After Deleting :",newObj.getSize())
print("Getting First and Rear")
print("First Element :",newObj.getFront())
print("Rear Element :",newObj.getRear())
```

Output Of The Program Is:

```
Displaying the Dequeue Before removeing
12 11 10 20 21 22
Size of list before Deleting 6
Getting First and Rear Before Deleting
First Element : 12
Rear Element : 22
Displaying the Dequeue After removeing 1st and last
11 10 20 21
Size of list After Deleting : 4
Getting First and Rear
First Element : 11
Rear Element : 21
```

Program 02:

```python
# Q2 Implement the Dequeue using Singly Linklist?
class  Node:
    def __init__(self ,data=None,next=None):
        self.data = data
        self.next=next
class linkList:
    def  __init__(self):
        self.head=None
        self.counter=0
    def isEmpty(self):
        return self.counter==0
    def insertAtFirst(self,data):
        newNode=Node(data,self.head)
        self.head=newNode
        self.counter+=1
    def insertAtLast(self,data):
        current=self.head
        newNode=Node(data)
        if not self.isEmpty():
            while(current.next!=None):
```

```
            current=current.next
        current.next=newNode
    else:
        self.head=newNode
    self.counter +=1
def deleteFromFirst(self):
    if self.isEmpty():
        print("The list is empty ")
    else:
        self.head=self.head.next
        self.counter-=1
def deleteFromLast(self):
    if self.isEmpty():
        print("The list is empty")
    else:
        current=self.head
        prevcurrent=None
        while(current.next!=None):
```

```
            prevcurrent=current
            current=current.next
        prevcurrent.next=None
        self.counter-=1
def getFront(self):
    if self.isEmpty():
        return ("The list is empty")
    else:
        return self.head.data
def getRear(self):
    if self.isEmpty():
        return ("The list is empty.")
    else:
        temp=self.head
        while(temp.next != None):
            temp = temp.next
        return temp.data
def getSize(self):
```

```
        if self.isEmpty():
            return 0
        else:
            return self.counter

    def show(self):
        if self.isEmpty():
            print("The list is empty")
        else:
            temp=self.head
            while(temp != None):
                print(temp.data,end=" ")
                temp = temp.next

# Testing the code
dll=linkList()
dll.insertAtFirst(4)
dll.insertAtFirst(22)
```

```
dll.insertAtLast(43)
dll.insertAtLast(34)
print("\nDisplaying the Dequeue Before removeing")
dll.show()
print("\nSize of list before Deleting",dll.getSize())
print("Getting First and Rear Before Deleting")
print("First Element :",dll.getFront())
print("Rear Element :",dll.getRear())
print("Displaying the Dequeue After removeing 1st and last")
dll.deleteFromFirst()
dll.deleteFromLast()
dll.show()
print("\nSize of list After Deleting : ",dll.getSize())
print("Getting First and Rear After Deleting")
print("First Element :",dll.getFront())
print("Rear Element :",dll.getRear())
```

Output Of The Above Program is:

```
Displaying the Dequeue Before removeing
22 4 43 34
Size of list before Deleting 4
Getting First and Rear Before Deleting
First Element : 22
Rear Element : 34
Displaying the Dequeue After removeing 1st and last
4 43
Size of list After Deleting :  2
Getting First and Rear After Deleting
First Element : 4
Rear Element : 43
```

Program 03:

```python
# Q3: Modify the print_queue() function in the example#2 lab#10 by printing the elements in the
list from rear to front.
class Node:
    def __init__(self, prev=None, data=None, next=None):
        self.data = data
        self.next = next
        self.prev = prev

class Deque:
    def __init__(self):
        self.front = None
        self.rear = None
        self.size = 0

    def insert_front(self, data):
        new_node = Node(None, data, self.front)
        if self.front is None:
            self.front = new_node
            self.rear = new_node
        else:
```

```
            self.front.prev = new_node
            self.front = new_node
        self.size += 1

    def insert_rear(self, data):
        new_node = Node(self.rear, data, None)
        if self.rear is None:
            self.front = new_node
            self.rear = new_node
        else:
            self.rear.next = new_node
            self.rear = new_node
        self.size += 1

    def shows(self):
        current = self.front
        while current:
            print(current.data, end=" ")
            current = current.next
    def print_queue(self):
        current = self.rear
        while current:
            print(current.data, end=" ")
            current = current.prev
deque_linked_list = Deque()
deque_linked_list.insert_front(42)
deque_linked_list.insert_front(62)
deque_linked_list.insert_rear(99)
deque_linked_list.insert_rear(76)
print("\nPrinting in Reverse Order")
deque_linked_list.print_queue()
print("\nDisplaying the Dequeue in Straight Order")
deque_linked_list.shows()
```

Output Of The Program Is:

```
Printing in Reverse Order
76 99 42 62
Displaying the Dequeue in Straight Order
62 42 99 76
```

Assignment 11

Program 01:

```python
#  Q1: Modify the example#1 lab#11 to display the output in reverse order i.e. 5 4 3 2 1. A recursive function should also be used in this program.
def naturalInReverse(n):
    if n == 0:
        return
    print(n,end=" ")
    naturalInReverse(n-1)

naturalInReverse(5)
```

Output Of Program Is:

```
5 4 3 2 1
```

Program 02:

```python
#  Q1: Modify the example#1 lab#11 to display the output in reverse order i.e. 5 4 3 2 1. A recursive function should also be used in this program.
def naturalInReverse(n):
    if n == 0:
        return
    print(n,end=" ")
    naturalInReverse(n-1)

naturalInReverse(5)
```

Output Of Program Is:

```
2 4 6 8 10
```

Program 03:

```python
# Q3 Write a program to display the sum of first N even numbers by using a recursive function.
def sum_even_numbers(n):
    if n == 0:
        return 0
    return 2*n + sum_even_numbers(n-1)

print(sum_even_numbers(5))
```

Output Of Program Is:

```
30
```

The

End