NAME : GINI CHACKO

ROLL : 8942

CLASS : SE COMPS B

BATCH : B

TOPIC : PYTHON EXPERIMENT 4

Aim: Write python program to

1) Write a program to sort a queue in python without using extra space

```
from queue import Queue

def minIndex(q, sortedIndex):
    min_index = -1
    min_val = 999999999999
    n = q.qsize()
    for i in range(n):
        curr = q.queue[0]
        q.get()

        if (curr <= min_val and i <= sortedIndex):
            min_index = i
            min_val = curr
        q.put(curr)
    return min_index

def insertMinToRear(q, min_index):
    min_val = None
    n = q.qsize()
    for i in range(n):
        curr = q.queue[0]
        q.get()
        if (i != min_index):
            q.put(curr)
        else:
            min_val = curr
    q.put(min_val)

def sortQueue(q):
    for i in range(1, q.qsize() + 1):
        min_index = minIndex(q, q.qsize() - i)
        insertMinToRear(q, min_index)

if __name__ == '__main__':
```

```
q = Queue()
n = int(input("Enter number of elements of list : "))
for i in range(0,n):
    b = int(input("Enter element : "))
    q.put(b)


sortQueue(q)

print("The sorted queue is : ")

while (q.empty() == False):
    print(q.queue[0], end = " ")
    q.get()
```

```
Enter number of elements of list : 6
Enter element : 9
Enter element : 5
Enter element : 8
Enter element : 1
Enter element : 6
Enter element : 3
The sorted queue is :
1 3 5 6 8 9
```

2) Write a python program to Implement shopping cart using linked-list. The user should be able to add item, remove item, display all the items and calculate total amount of the cart. Each Item details contains (Item name, quantity, price).

```
class Node:

    def __init__(self, name,quantity, price):
        self.name = name
        self.quantity=quantity
        self.price=price
        self.next = None

class LinkedList:

    def __init__(self):
        self.head = None

    def append_product(self, new_data):
        if self.head is None:
            self.head = new_data
            return
        product = self.head
        while(product.next):
            product = product.next
```

```
        product.next = new_data


    def delete_product(self, rem_data):
        product=self.head
        if (product is not None):
            if (product.name == rem_data):
                self.head = product.next
                product = None
                return
        while (product is not None):
            if product.name == rem_data:
                break
            prev = product
            product = product.next
        print(rem_data,"is removed from the shopping cart")
        prev.next = product.next
        product = None


    def print_product(self):

        product = self.head
        print("\n-------SHOPPING CART LIST-------")

        while (product):
            print("\n\nProduct Name :", product.name, "\nQuantity : ", product.quantity, "\nP
            product = product.next

    def get_Total(self):
        product = self.head
        total_cost = 0
        while product:
            total_cost = total_cost + (product.quantity * product.price)
            product = product.next
        return total_cost

if __name__ == '__main__':

  print("==============WELCOME TO YOUR SHOPPING CART==============")

  obj = LinkedList()
  choice = 1



  while choice != 5:
      print( )
      print("************MENU**************")
      print("1. Add an item ")
      print("2. Delete an item ")
      print("3. Display items in cart")
```

```
        print("4. Total amount ")
        print("5. Exit the system ")
        choice = int(input("\nEnter choice: "))

        if choice == 1:
            product_name = input("Enter product name : ")
            product_qty = int(input("Enter quantity : "))
            product_price = float(input("Enter price of the product : Rs."))

            new_product = Node(product_name, product_qty, product_price)
            obj.append_product(new_product)

            print("\nYour item %s is successfully added" %(product_name) )

        elif choice == 2:
            remove_product = input("Enter Product Name: ")
            obj.delete_product(remove_product)

            print("\nYour item %s is successfully deleted" %(remove_product) )

        elif choice == 3:
            obj.print_product()

        elif choice == 4:
            total_cost = obj.get_Total()
            print("\nTotal Amount : Rs.%s" %(total_cost))

        elif choice == 5:
            print("\nExiting!")

        else:
            print("\nInvalid choice!!")
```

```
Enter choice: 4

Total Amount : Rs.600.0

**********MENU**********
1. Add an item
2. Delete an item
3. Display items in cart
4. Total amount
5. Exit the system

Enter choice: 2
Enter Product Name: onion
onion is removed from the shopping cart

Your item onion is successfully deleted

**********MENU**********
1. Add an item
2. Delete an item
```

```
3. Display items in cart
4. Total amount
5. Exit the system

Enter choice: 3

-------SHOPPING CART LIST-------


Product Name : cabbage
Quantity :  2
Price :  50.0


Product Name : wheat
Quantity :  1
Price :  100.0

**********MENU**********
1. Add an item
2. Delete an item
3. Display items in cart
4. Total amount
5. Exit the system

Enter choice: 4

Total Amount : Rs.200.0

**********MENU**********
1. Add an item
2. Delete an item
3. Display items in cart
4. Total amount
5. Exit the system


Enter choice: 5

Exiting!
```

POSTLABS (EXPERIMENT – 4)

1. **What is Doubly link list. Write a python program to create a doubly linked list and add a new node after a given node , traverse a doubly linked list.**

Ans :

Doubly Linked List is a variation of the linked list. The linked list is a linear data structure which can be described as the collection of nodes. Nodes are connected through pointers. Each node contains two fields: data and pointer to the next field. The first node of the linked list is called the head, and the last node of the list is called the tail of the list.

One of the limitations of the singly linked list is that it can be traversed in only one direction that is forward. The doubly linked list has overcome this limitation by providing an additional pointer that points to the previous node. With the help of the previous pointer, the doubly linked list can be traversed in a backward direction thus making insertion and deletion operation easier to perform. So, a typical node in the doubly linked list consists of three fields:

- **Data** represents the data value stored in the node.
- **Previous** represents a pointer that points to the previous node.
- **Next** represents a pointer that points to the next node in the list.

CODE:

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

    def push(head_ref, new_data):

        new_node = Node(new_data)

        new_node.data = new_data

        new_node.next = head_ref
        new_node.prev = None
```

```
    if (head_ref != None):
        head_ref.prev = new_node

        head_ref = new_node

    return head_ref

def insertBefore(head_ref, next_node, new_data):

    if (next_node == None):
        print("the given next node cannot be NULL")
        return

    new_node = Node(new_data)

    new_node.prev = next_node.prev

    next_node.prev = new_node

    new_node.next = next_node

    if (new_node.prev != None):
        new_node.prev.next = new_node

    else:
        head_ref = new_node

    return head_ref

def printList(node):
    last = None
    print("Traversal in forward direction ")
    while (node != None):
        print(node.data, end=" ")
        last = node
        node = node.next

    print("\nTraversal in reverse direction ")
    while (last != None):
```

```
        print(last.data, end=" ")
        last = last.prev


if __name__ == '__main__':

    head = None
    head = push(head, 7)

    head = push(head, 1)

    head = push(head, 4)

    head = insertBefore(head, head.next, 8)

    print("Created Doubly Linked List is: ")
    printList(head)
```

OUTPUT:

```
Created Doubly Linked List is:
Traversal in forward direction
4 3 2 6
Traversal in reverse direction
6 2 3 4
```

2. Write a python program to implement linked list using collection.dequeue().

Ans:

CODE :

```
import collections

linked_lst = collections.deque()

linked_lst.append('1')
linked_lst.append('2')
linked_lst.append('3')

print("Elements in the linked_list : ")
print(linked_lst)

linked_lst.insert(1,'4')

print("Elements in the linked_list : ")
print(linked_lst)

linked_lst.pop()

print("Elements in the linked_list : ")
print(linked_lst)

linked_lst.remove('4')

print("Elements in the linked_list : ")
print(linked_lst)
```

OUTPUT :

```
Elements in the linked_list :
deque(['1', '2', '3'])
Elements in the linked_list :
deque(['1', '4', '2', '3'])
Elements in the linked_list :
deque(['1', '4', '2'])
Elements in the linked_list :
deque(['1', '2'])
```