**Task # 1:** Design & implement all methods of Simple Queue.

**Solution**

using System;

namespace LAb8\_Queue

{

internal class queue

{

int[] a = new int[5];

int front = 0;

int rear = -1;

internal bool isempty()

{

if (front == -1 && rear == -1)

{

return true;

}

else

{

return false;

}

}

internal bool isfull()

{

if (rear == a.Length - 1)

{

Console.WriteLine("Queue Is full");

return true;

}

else

{

return false;

}

}

internal int Enqueue(int data)

{

if (isfull())

{

return 0;

}

rear = rear + 1;

a[rear] = data;

Console.WriteLine("{0} Inserted ", data);

return 1;

}

internal void peak()

{

if (!isempty())

{

Console.WriteLine("Queue Peak Element:{0}", a[front]);

}

else

{

Console.WriteLine("Queue Is Empty");

}

}

internal int dequeue()

{

if (isempty())

{

return 0;

}

int data = a[front];

front = front + 1;

Console.WriteLine("Dequeue deleted Element:{0}", data);

return 1;

}

internal void count()

{

int count = 0;

if (!isempty())

{

for (int i = front; i <= rear; i++)

{

count++;

}

Console.WriteLine("Total Element In Queue Are {0}", count);

}

else

{

Console.WriteLine("Queue Is Empty");

}

}

internal void Display()

{

if (!isempty())

{

Console.WriteLine("Total Element In Queue Are ");

for (int i = front; i <= rear; i++)

{

Console.WriteLine($"{a[i]}");

}

}

else

{

Console.WriteLine("Queue Is Empty");

}

}

static void Main(string[] args)

{

queue q = new queue();

int res;

do

{

Console.WriteLine("Please Choose From Below");

Console.WriteLine(" 1 ) Enqueue");

Console.WriteLine(" 2 ) Dequeue");

Console.WriteLine(" 3 ) Isfull method");

Console.WriteLine(" 4 ) Isempty method");

Console.WriteLine(" 5 ) peak");

Console.WriteLine(" 6 ) count");

Console.WriteLine(" 7 ) Display");

Console.Write("Enter : ");

res = int.Parse(Console.ReadLine());

if (res == 1)

{

Console.Write("Please Enter Data : ");

q.Enqueue(Convert.ToInt32(Console.ReadLine()));

}

else if (res == 2)

{

q.dequeue();

}

else if (res == 3)

{

if (q.isfull())

{

Console.WriteLine("Queue Is Full");

}

}

else if (res == 4)

{

if (q.isempty())

{

Console.WriteLine("Queue Is Empty");

}

}

else if (res == 5)

{

q.peak();

}

else if (res == 6)

{

q.count();

}

else if (res == 7)

{

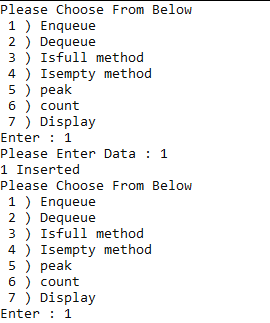
q.Display();

}

} while (res != 1 || res != 2 || res != 3 || res != 4 || res != 5 || res != 6 || res != 7);}}}

**Text

Description automatically generatedOutput**

****

**Task # 2** Write a code to implement messaging system that is capable to handle message requests, message capacity must be “5” and after that store messages in buffer. Methods to implement:

* Enqueue
* Dequeue
* Print Queue

**Solution**

using System;

using System.Collections;

namespace LAb8\_Queue

{

internal class queue

{

static int max = 5;

int[] per = new int[max];

public ArrayList buffer = new ArrayList();

int front = -1;

int rear = -1;

public bool isempty()

{

if (rear == -1 && front == -1)

{

return true;

}

return false;

}

public bool isfull()

{

if (rear == per.Length - 1)

{

return true;

}

return false;

}

public void enqueue(int data)

{

if (!isfull())

{

if (isempty())

{

front = 0;

rear = 0;

per[rear] = data;

}

else

{

rear = rear + 1;

per[rear] = data;

}

}

else

{

Console.WriteLine("Queue Is Full ");

buffer.Add(data);

}

}

public void dequeue()

{

if (!isempty())

{

if (front == rear)

{

front = rear = -1;

}

else

{

for (int i = front; i < rear; i++)

{

per[i] = per[i + 1];

}

rear = rear - 1;

if (buffer.Count != 0)

{

rear = rear + 1;

per[4] = Convert.ToInt32(buffer[0]);

buffer.RemoveAt(0);

}

}

}

else

{

Console.WriteLine("Queue Is Empty!!! ");

Console.WriteLine();

}

}

public void print()

{

int count1 = 0;

for (int i = front; i <= rear; i++)

{

Console.WriteLine(per[i]);

}

}

public void printarrlist()

{

for (int i = 0; i < buffer.Count; i++)

{

Console.WriteLine(buffer[i]);

}

}

static void Main(string[] args)

{

queue queue = new queue();

int res;

do

{

Console.WriteLine("---------------------------------------------------------------------------------------------");

Console.WriteLine(" 1) Enque");

Console.WriteLine(" 2) Deqeue");

Console.WriteLine(" 3) Print Queue");

Console.WriteLine(" 4) Print ArrayList");

Console.WriteLine("\n\n---------------------------------------------------------------------------------------------");

Console.Write("Enter : ");

res = Convert.ToInt32(Console.ReadLine());

if (res == 1)

{

Console.Write("Enter Your Data : ");

queue.enqueue(Convert.ToInt32(Console.ReadLine()));

}

else if (res == 2)

{

queue.dequeue();

Console.WriteLine("DeQueue operation Is Exeuated ");

}

else if (res == 3)

{

Console.WriteLine("---------------------------------------------------------------------------------------");

queue.print();

}

else if (res == 4)

{

Console.WriteLine("---------------------------------------------------------------------------------------");

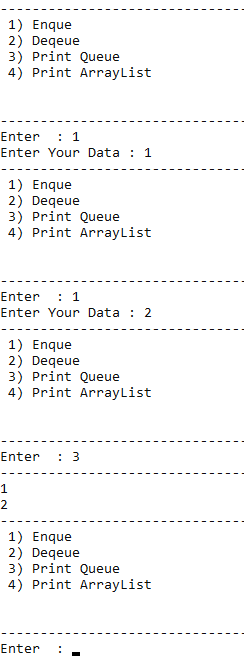
queue.printarrlist();

}

} while (res != 1 || res != 2 || res != 3 || res != 4);}}}

**Text

Description automatically generatedOutput**

****

**Solution With Linklist**

using System;

namespace ConsoleApp27

{

## NODE CLASS

public class Node

{

public String Data; public Node prev, next; public int count;

public Node(string dt)

{

Data = dt; next = null; prev = null;

}

}

## LINKEDLIST CLASS

class LinkedList

{

public Node head; public int counter;

public void Enqueue(string msg)

{

counter++;

Node new\_node = new Node(msg); if (head == null)

head = new\_node;

else

{

Node temp; temp = head;

while (temp.next != null) temp = temp.next;

temp.next = new\_node; new\_node.prev = temp;

}

}

public void Dequeue()

{

counter--;

if (head == null) return;

if (head.next == null) head.Data = null;

if (head.next != null)

{

head = head.next; head.prev.next = null; head.prev = null;

}

}

public void Print()

{

Node temp; temp = head; int i = 1;

while (temp != null && i <= 5)

{

Console.WriteLine("\tMESSAGE # {0} : " + temp.Data, i++); temp = temp.next;

}

}

}

## MAIN CLASS

class Program

{

static void Main(string[] args)

{

LinkedList Messages = new LinkedList();

LinkedList Buffer = new LinkedList();

Messages.Enqueue("Hello");

Messages.Enqueue("How Are You?");

Messages.Enqueue("Kese Ho?");

Messages.Enqueue("Tabiyat Kesi Hy?");

while (true)

{

Console.Clear();

Console.WriteLine(" ");

Console.WriteLine("\* \* \* WELCOME TO MESSAGE MANAGEMENT SYSTEM \* \* \*");

Console.WriteLine(" ");

Console.WriteLine("\n\t1.Enqueue");

Console.WriteLine("\t2.Dequeue");

Console.WriteLine("\t3.Print Queue");

Console.WriteLine("\n ");

Console.Write(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Option : ");

int resp = Convert.ToInt32((Console.ReadLine()));

Console.WriteLine(" ");

if (resp == 1)

{

Console.WriteLine("\n\n ");

Console.Write("\tEnter Message : ");

string newmessage = Console.ReadLine();

if (Messages.counter <= 5)

{

Messages.Enqueue(newmessage);

if (Messages.head.Data == null && Messages.head.next != null)

{

Messages.head = Messages.head.next;

}

}

if (Messages.counter > 5)

{

Buffer.Enqueue(newmessage);

Console.WriteLine(" ");

}

}

else if (resp == 2)

{

Messages.Dequeue();

if (Messages.counter <= 5 && Buffer.head != null)

{

if (Buffer.head.next == null)

{

Buffer.head.next = null;

Buffer.head.prev = null;

Buffer.head.Data = null;

}

if (Buffer.head.next != null)

{

Messages.Enqueue(Buffer.head.Data);

Buffer.head = Buffer.head.next;

}

else

{

continue;

}

}

}

else if (resp == 3)

{

if (Messages.head.Data == null)

{

Console.WriteLine("Messages List is Empty");

}

else

{

Messages.Print();

}

}

else

{

Console.WriteLine("\n ");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INVALID INPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine(" ");

}

Console.WriteLine("\nPress Enter Key to Continue...");

Console.ReadKey();

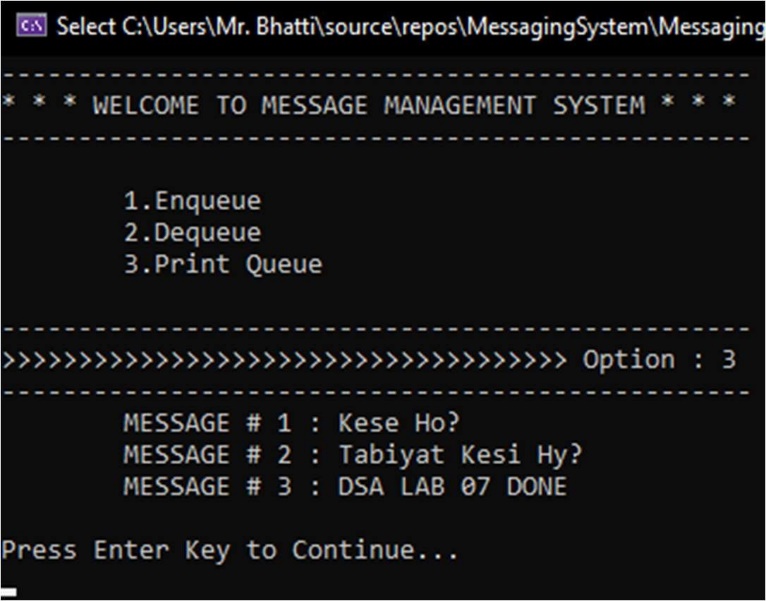
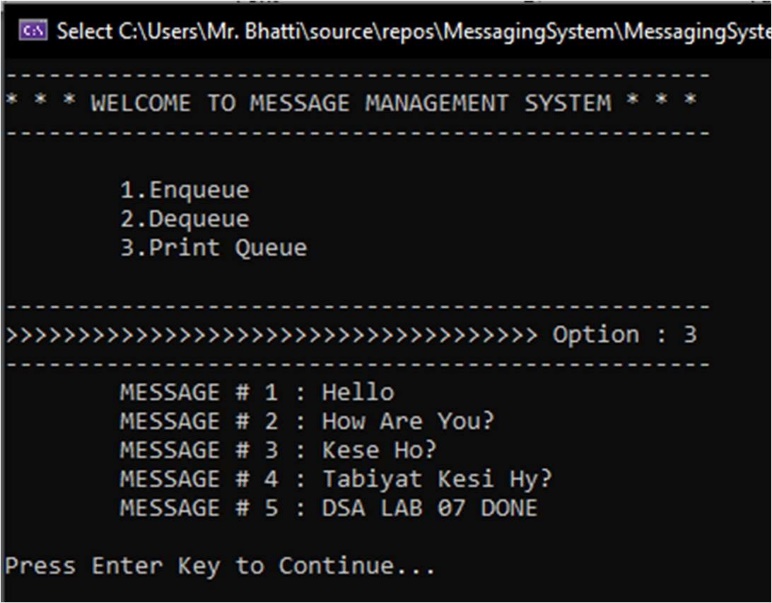
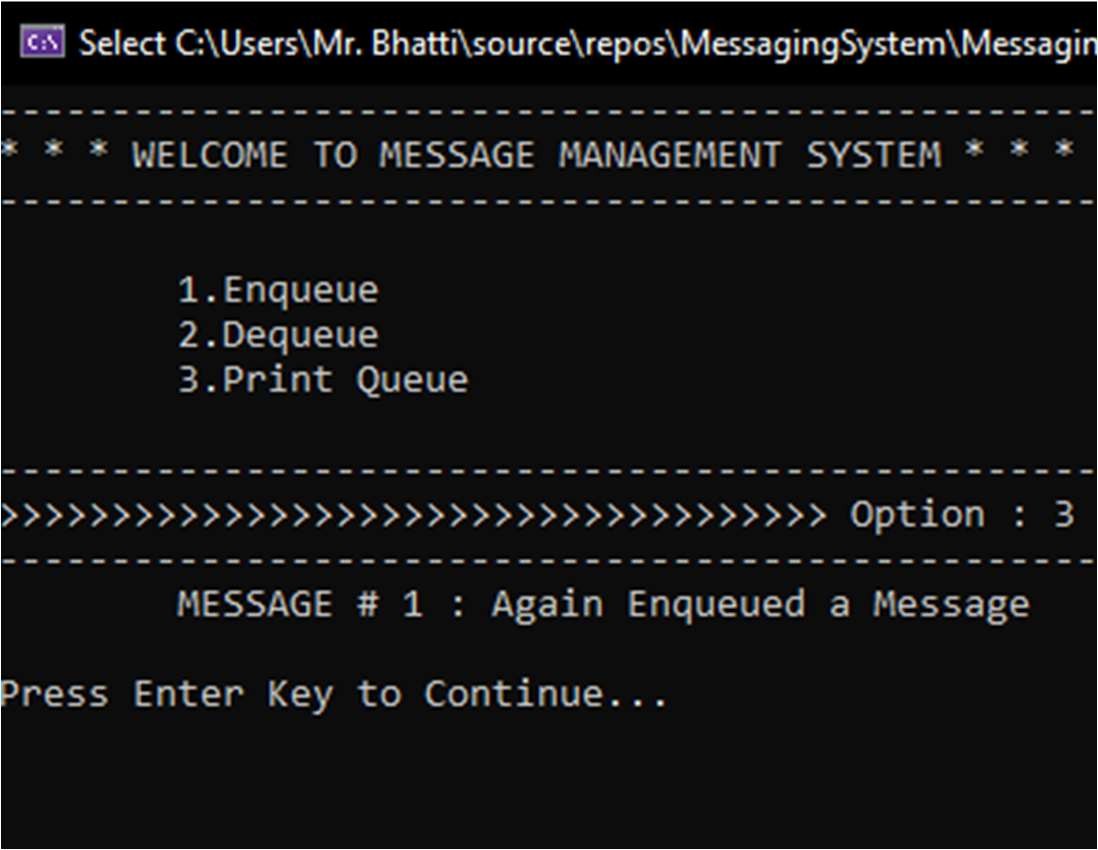
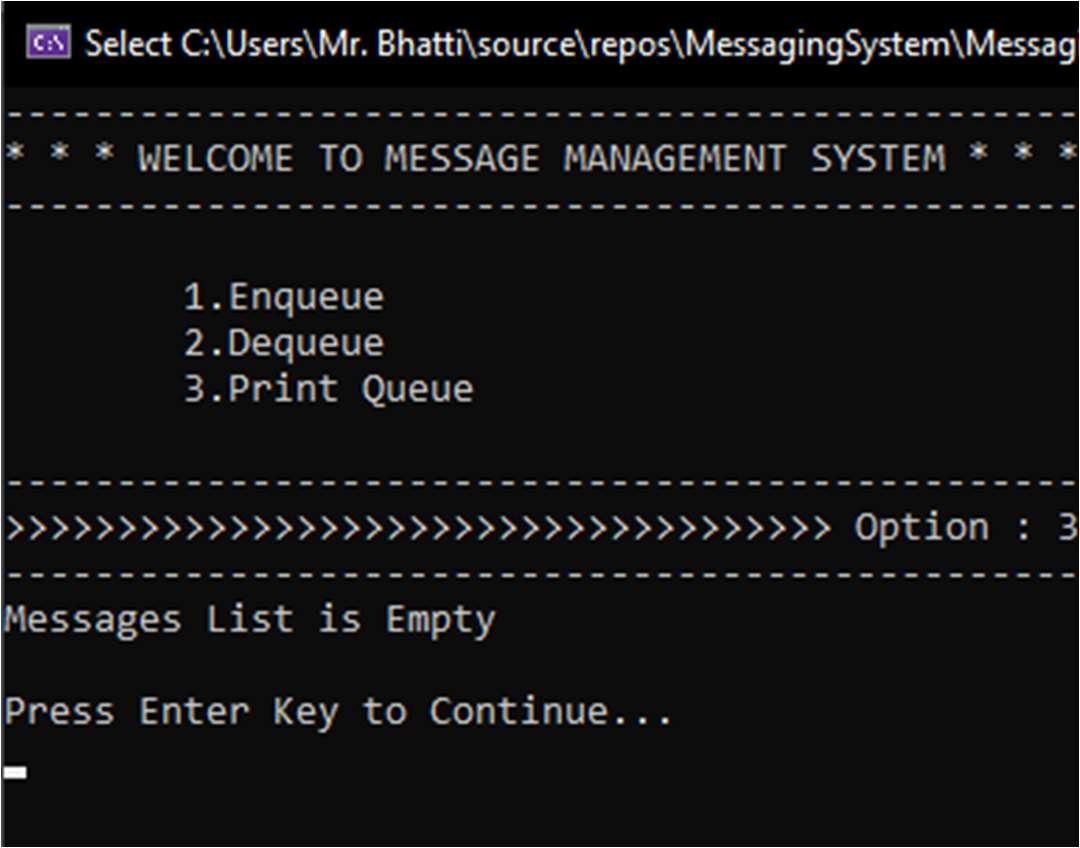
}

}

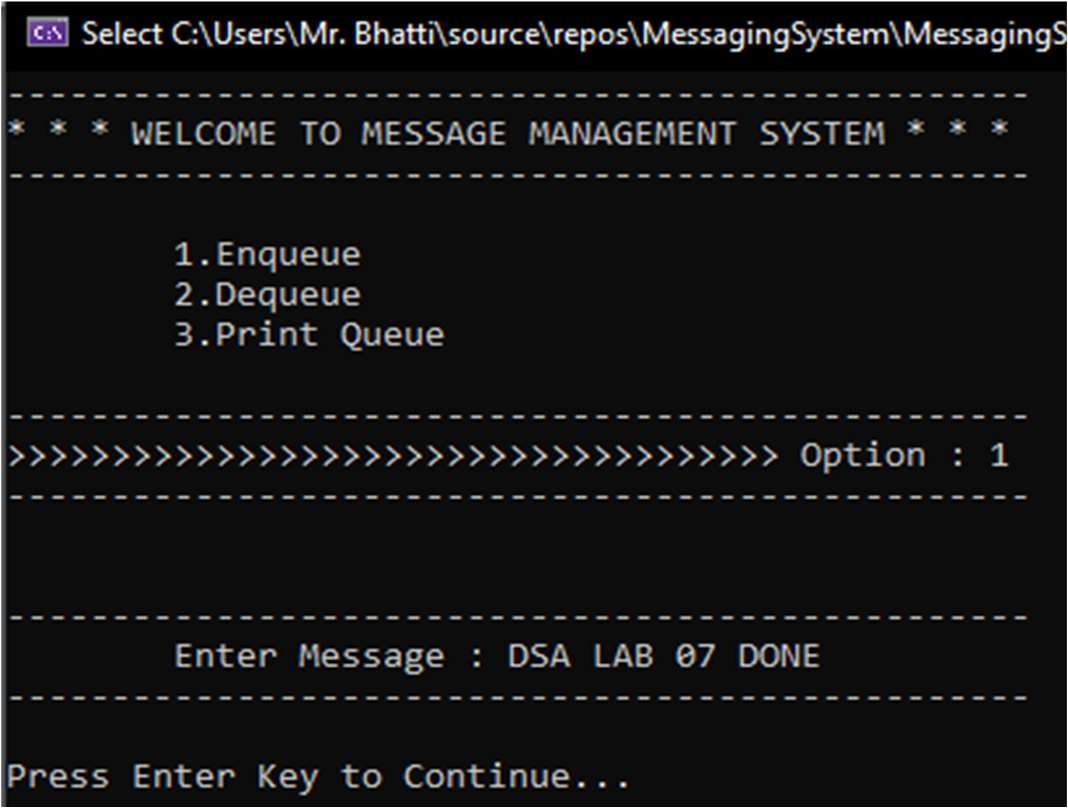
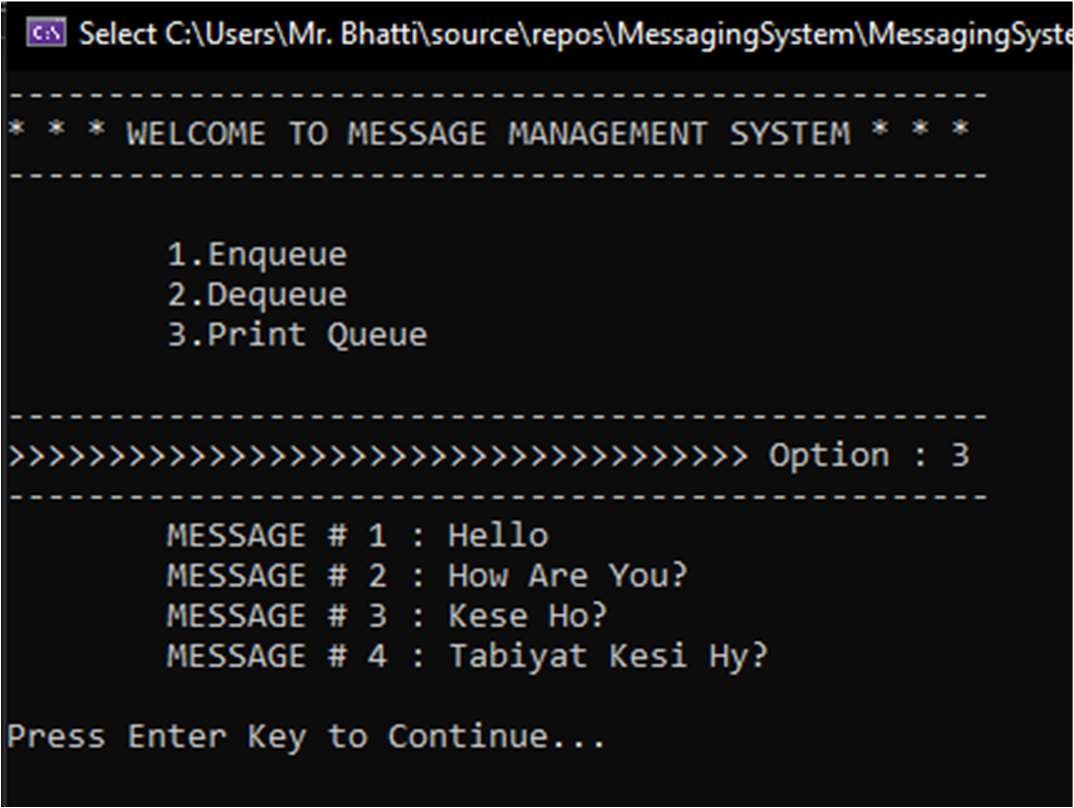
}

}

**Output**



*Figure 1 : Printing Queue Figure 2 : Enqueuing Message in Queue*



*Figure 3 : Again Printing Queue Figure 4 : Dequeuing 2 Messages*

*Figure 5 : After Dequeuing all messages Figure 6 : Again Enqueuing Messages*

**Task # 3:** Design & implement all methods of Circular Queue

**Solution**

using System;

namespace LAb8\_Queue

{ internal class queue

{ int[] a = new int[5];

int front = 0;

int temp = 0;

int rear = 0;

internal bool isempty()

{

if (front == rear)

{return true;}

else

{return false;}}

internal bool isfull()

{

if ((rear + 1) % a.Length == front)

{return true;}

else

{return false;}}

internal void peak()

{

if (isempty())

{

Console.Write("Queue is Empty");

return;

}

if (temp == 0)

{

Console.WriteLine($"Peak Element Is {a[temp + 1]}");}

else

{

temp = (temp + 1) % a.Length;

Console.WriteLine($"Peak Element Is {a[temp]}");} }

internal void count()

{

int count = 0;

if (isempty())

{

Console.Write("Queue is Empty");return;}

for (int i = (front + 1)%a.Length; i <= rear; i++)

{count++;}

Console.WriteLine($"Total Element In Queue Are {count}");}

internal void Enqueue(int data)

{

if (isfull())

{

Console.WriteLine("Circular Queue Is full");

return;}

rear = (rear + 1) % a.Length;

a[rear] = data;

Console.WriteLine("{0} Inserted ", data);return;}

internal void dequeue()

{

if (isempty())

{

Console.WriteLine("Circular Queue Is Empty");return;

}

front = (front + 1) % a.Length;

int data = a[front];

temp = front;

Console.WriteLine("Circular Dequeue deleted:{0}", data);

return;

}

public void display()

{

if (isempty())

{

Console.Write("Queue is Empty");

return;

}

Console.Write("Elements in the circular queue are: ");

for (int i = (front + 1) %a.Length; i <= rear; i++)

{

Console.Write(a[i]);

Console.Write(" ");

}}

static void Main(string[] args)

{ queue c = new queue();

int res;

do

{

Console.WriteLine("\nPlease Choose From Below");

Console.WriteLine(" 1 ) Enqueue");

Console.WriteLine(" 2 ) Dequeue");

Console.WriteLine(" 3 ) Isfull method");

Console.WriteLine(" 4 ) Isempty method");

Console.WriteLine(" 5 ) peak");

Console.WriteLine(" 6 ) count");

Console.WriteLine(" 7 ) Display");

Console.Write("Enter : ");

res = int.Parse(Console.ReadLine());

if (res == 1)

{

Console.Write("Please Enter Data : ");

c.Enqueue(Convert.ToInt32(Console.ReadLine()));}

else if (res == 2)

{ c.dequeue(); }

else if (res == 3)

{

if (c.isfull())

{

Console.WriteLine("Queue Is Full");}}

else if (res == 4)

{

if (c.isempty())

{

Console.WriteLine("Queue Is Empty");}}

else if (res == 5)

{ c.peak(); }

else if (res == 6)

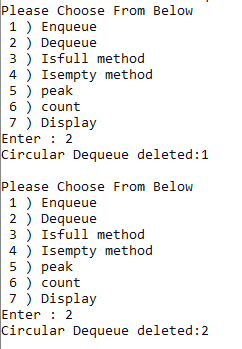
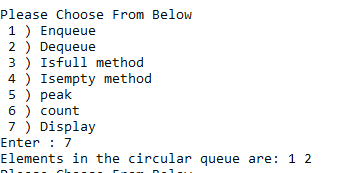
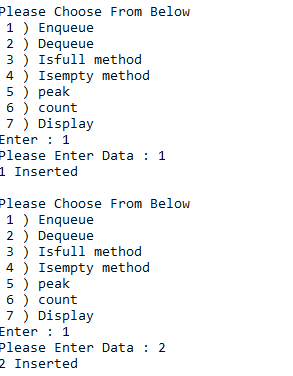
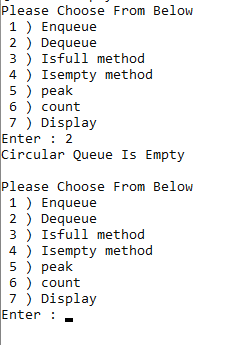
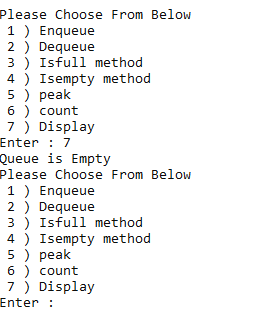
{ c.count(); }

else if (res == 7)

{ c.display(); }

} while (res != 1 || res != 2 || res != 3 || res != 4 || res != 5 || res != 6 || res != 7);}}}

**Output**



**Task # 4:** Design and implement for Priority Queue.

Method 1: Ordering in/ after Enqueue method

Method 2: Separate queues for different priorities.

**Solution**

using System;

namespace LAb8\_Queue

{

class Node

{

public int priority;

public int info;

public Node link;

public Node(int i, int pr)

{

info = i;

priority = pr;

link = null;

}

}

class PriorityQueueL

{

private Node front;

public PriorityQueueL()

{

front = null;

}

public void Insert(int element, int elementPriority)

{

Node temp, p;

temp = new Node(element, elementPriority);

if (IsEmpty() || elementPriority < front.priority)

{

temp.link = front;

front = temp;

}

else

{

p = front;

while (p.link != null && p.link.priority <= elementPriority)

p = p.link;

temp.link = p.link;

p.link = temp;

}}

public int Delete()

{

int element;

if (IsEmpty())

{

throw new System.InvalidOperationException("Queue Underflow");

}

else

{

element = front.info;

front = front.link;

}

return element;

}

public bool IsEmpty()

{

return (front == null);

}

public void Display()

{

Node p = front;

if (IsEmpty())

{

Console.WriteLine("Queue is empty\n");

}

else

{

Console.WriteLine("Queue is :");

Console.WriteLine("Element Priority");

while (p != null)

{

Console.WriteLine(p.info + " \t " + p.priority);

p = p.link;}

}

Console.WriteLine("");}}

internal class queue

{

static void Main(string[] args)

{

int res, element, elementPriority;

PriorityQueueL pq = new PriorityQueueL();

do

{

Console.WriteLine("1.Insert a new element");

Console.WriteLine("2.Delete an element");

Console.WriteLine("3.Display the queue");

Console.WriteLine("4.Quit");

Console.Write("Enter your choice : ");

res = Convert.ToInt32(Console.ReadLine());

if (res == 1)

{

Console.Write("Enter the element to be inserted: ");

element = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter its priority : ");

elementPriority = Convert.ToInt32(Console.ReadLine());

pq.Insert(element, elementPriority);

}

else if (res == 2)

{ Console.WriteLine("Deleted element is: " + pq.Delete()); }

else if (res == 3)

{ pq.Display(); }

else if (res == 4)

{ break; }

} while (res != 1 || res != 2 || res != 3 || res != 4);}}}

**Output**

