**LAB # 08**

**Task # 01: Design & implement all methods of Simple Queue.**

**Solution:**

namespace Queue

{

public class SimpleQueue

{

private int[] a;

private int front = -1, rear = -1;

private int length;

public SimpleQueue(int l)

{

length = l;

a = new int[length];

}

public bool isEmpty()

{

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

{

Console.WriteLine("Queue is Empty");

return true;

}

else

{

return false;

}

}

public bool isFull()

{

if (rear == (length-1))

{

Console.WriteLine("Queue is Full");

return true;

}

else

{

return false;

}

}

public void Enqueue(int x)

{

if (isFull())

{

return;

}

else if (isEmpty())

{

rear = 0;

front = 0;

a[rear] = x;

}

else

{

rear++;

a[rear] = x;

}

}

public void Dequeue()

{

if (isFull())

{

front += 1;

return;

}

else if (isEmpty())

{

rear = -1;

front = -1;

}

else

{

front = +1;

}

}

public void Display()

{

if (isEmpty())

{

}

else

{

if (front < length)

{

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

{

Console.Write(a[i] + " ");

}

}

else

{

Console.WriteLine("Empty Queue");

}

}

Console.ReadKey();

}

public bool Menu()

{

Console.Clear();

Console.Write("1)Enqueue\n2)Dequeue\n3)Display Queue\n4)Exit\nChoose:");

switch (Console.ReadLine())

{

case "1":

Console.WriteLine("Enter value to Enqueue:");

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

Enqueue(n);

return true;

case "2":

Dequeue();

return true;

case "3":

Display();

return true;

case "4":

return false;

default:

return true;

}

}

public static void Main(string[] arg)

{

Console.WriteLine("Enter Queue length");

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

SimpleQueue sq = new SimpleQueue(n);

bool m = true;

while (m)

{

m = sq.Menu();

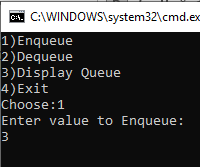
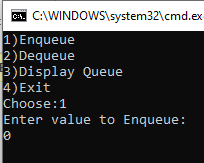
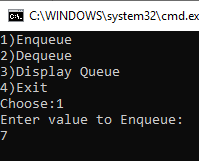
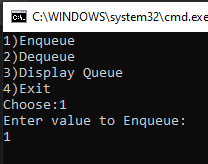
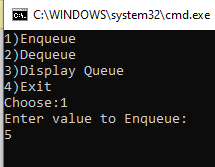
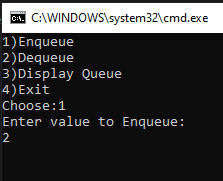
}

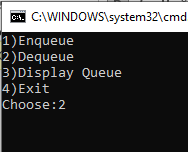
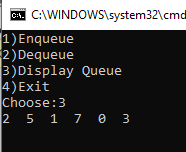
}

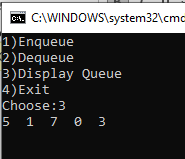
}

}

**Output:**







**Task # 02: Design & implement all methods of Circular Queue.**

**Solution:**

public class CircularQueue

{

private int rear, front, length;

int[] a;

CircularQueue(int l)

{

front = -1;

rear = -1;

length = l;

a = new int[length];

}

public bool IsFull()

{

if ((front == 0 && rear == length - 1) ||(rear == (front - 1) % (length - 1)))

{

return true;

}

else

{

return false;

}

}

public bool IsEmpty()

{

if (front == -1)

{

return true;

}

else

{

return false;

}

}

public void Enqueue(int x)

{

if (IsFull())

{

Console.WriteLine("Queue is full");

Console.ReadKey();

return;

}

else if (front == -1)

{

front = rear = 0;

a[rear] = x;

}

else if (rear == length - 1 && front != 0)

{

rear = 0;

a[rear] = x;

}

else

{

rear++;

a[rear] = x;

}

}

public void Dequeue()

{

if (IsEmpty())

{

Console.WriteLine("Queue is Empty");

Console.ReadKey();

return;

}

else if (rear == front)

{

front = -1;

rear = -1;

}

else if (front == length - 1)

{

front = 0;

}

else

{

front++;

}

}

public void Display()

{

if (IsEmpty())

{

Console.WriteLine("Queue is Empty");

Console.ReadKey();

return;

}

if (rear >= front)

{

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

{

Console.Write(a[i] + " ");

}

}

else if (rear < front)

{

for (int i = front; i <= length-1; i++)

{

Console.Write(a[i] + " ");

}

for(int i = 0; i <= rear; i++)

{

Console.Write(a[i] + " ");

}

}

else

{

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

{

Console.Write(a[i] + " ");

}

for (int i = 0; i <= rear; i++)

{

Console.Write(a[i] + " ");

}

}

Console.ReadKey();

}

public bool Menu()

{

Console.Clear();

Console.Write("1)Enqueue\n2)Dequeue\n3)Display Queue\n4)Exit\nChoose:");

switch (Console.ReadLine())

{

case "1":

Console.WriteLine("Enter value to Enqueue:");

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

Enqueue(n);

return true;

case "2":

Dequeue();

return true;

case "3":

Display();

return true;

case "4":

return false;

default:

return true;

}

}

public static void Main(string[] arg)

{

Console.WriteLine("Enter Queue length");

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

CircularQueue cq = new CircularQueue(n);

bool m = true;

while (m)

{

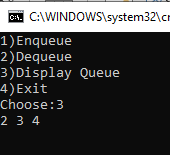
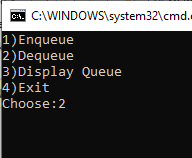
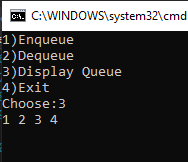
m = cq.Menu();

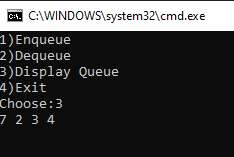
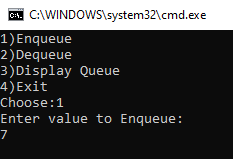
}

}

}

**Output:**





**Task # 03: Design & implement Priority Queue.**

**Solution:**

public class PriorityQueue

{

private int[] a;

private int front;

private int rear;

private int length;

private int count;

public PriorityQueue(int l)

{

a = new int[l];

front = 0;

rear = -1;

length = l;

count = 0;

}

public bool IsFull()

{

if (rear == (length-1))

{

return true;

}

else

{

return false;

}

}

public bool IsEmpty()

{

if (front==-1)

{

return true;

}

else

{

return false;

}

}

public void Enqueue(int item)

{

if (!IsFull())

{

a[++rear] = item;

count++;

}

else

{

Console.WriteLine("Queue Overflow");

}

}

public int Dequeue()

{

if (!IsEmpty())

{

count--;

int idx = 0;

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

{

if (Priority() == a[i])

{

idx = i;

}

}

int de = a[idx];

a[idx] = a[rear];

rear--;

return de;

}

else

{

Console.WriteLine("Queue is Empty");

return -1;

}

}

public int Priority()

{

int P = a[front];

for (int i = 0; i <= rear; i++)

{

if (P > a[i])

{

P = a[i];

}

}

return P;

}

public void Display()

{

if (IsEmpty())

{

Console.WriteLine("Queue is Empty");

return;

}

else

{

Console.WriteLine("Items in Queue:");

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

{

Console.WriteLine("{0} ", a[i]);

}

Console.WriteLine();

}

}

public static void Main(string[] arg)

{

PriorityQueue Q = new PriorityQueue(5);

Q.Enqueue(2);

Q.Enqueue(1);

Q.Enqueue(6);

Q.Enqueue(5);

Q.Enqueue(3);

Q.Display();

Console.WriteLine("Dequeuing:");

Q.Dequeue();

Console.WriteLine();

Q.Display();

Q.Enqueue(10);

Q.Dequeue();

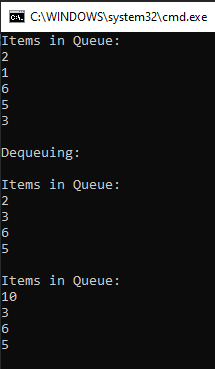
Q.Display();

Console.ReadLine();

}

}

**Output:**



**Output:**