Q: - Write a program to create a doubly linked list. Implement any sorting algorithm on the list and search for the user asked value.

Input:-

Class Dlinklist:-

class Dlinklist

{

DNode start;

public Dlinklist()

{

start = new DNode();

}

public bool Underflow()

{

if (start.next==null)

{

return true;

}

else

{

return false;

}

}

public bool InsertAtBeg(int val)

{

DNode n = new DNode(val);

if (!Underflow())

{

n.next = start.next;

start.next.previous = n;

start.next = n;

return true;

}

else

start.next = n;

return true;

}

public bool InsertAtEnd(int val)

{

DNode n = new DNode(val);

if (!Underflow())

{

DNode temp = start.next;

while (temp.next!=null)

{

temp = temp.next;

}

temp.next = n;

n.previous = temp;

return true;

}

else

{

start.next = n;

return true;

}

}

public bool InsertAfter(int oldval,int newval)

{

DNode n = new DNode(newval);

if (!Underflow())

{

DNode temp = start.next;

while (temp.data==oldval||temp.next!=null)

{

if (temp.data==oldval)

{

if (temp.next!=null)

{

n.next = temp.next;

n.previous = temp;

temp.next.previous = n;

temp.next = n;

return true;

}

else

{

n.next = temp.next;

n.previous = temp;

temp.next = n;

return true;

}

}

else

{

temp = temp.next;

}

}

if (temp.next==null)

{

Console.WriteLine("Value not found so Insert at End");

InsertAtEnd(newval);

return true;

}

}

start.next = n;

return true;

}

public bool DelFirst()

{

if (!Underflow())

{

if (start.next.next!=null)

{

start.next = start.next.next;

start.next.previous = null;

return true;

}

else

{

start.next = null;

return true;

}

}

else

return false;

}

public bool DelEnd()

{

if (!Underflow())

{

DNode temp = start.next;

while (temp.next.next != null)

{

temp = temp.next;

}

temp.next = null;

return true;

}

else

{

start.next = null;

return true;

}

}

public bool DelAfter(int oldval)

{

DNode n = new DNode(oldval);

if (!Underflow())

{

DNode temp = start.next;

while (temp.data == oldval || temp.next != null)

{

if (temp.data == oldval)

{

if (temp.next.next != null)

{

temp.next = temp.next.next;

temp.next.previous = n;

return true;

}

else

{

temp.next = null;

return true;

}

}

temp = temp.next;

}

if (temp.data != oldval)

{

Console.WriteLine("Value not found...!!");

return true;

}

}

return false;

}

public void display()

{

if (!Underflow())

{

DNode temp = start.next;

Console.WriteLine("Your List:");

while (temp.next!=null)

{

Console.Write(temp.data+" ");

temp = temp.next;

}

Console.WriteLine(temp.data+" ");

}

else

{

Console.WriteLine("List is Empty");

}

}

public void PreDisplay(int val)

{

if (!Underflow())

{

DNode n = new DNode(val);

DNode temp = start.next;

while (temp.data==val||temp.next!=null)

{

if (temp.data==val)

{

if (temp.next == null)

{

Console.WriteLine("Previous =" + temp.previous.data);

Console.WriteLine("Next = null");

break;

}

else if (temp.previous == null)

{

Console.WriteLine("Previous = null");

Console.WriteLine("Next = " + temp.next.data);

break;

}

Console.WriteLine("Previous ="+temp.previous.data);

Console.WriteLine("Next = " + temp.next.data);

break;

}

temp = temp.next;

}

}

else

{

Console.WriteLine("Underflow");

}

}

Class Node:-

class DNode

{

internal DNode next;

internal DNode previous;

internal int data;

public DNode()

{

next = null;

previous = null;

data = 0;

}

public DNode(int val)

{

data = val;

}

}

Main:-

Dlinklist obj = new Dlinklist();

while (true)

{

Console.WriteLine("What you want to Perform:");

Console.WriteLine("1-Insert at Beg\n2-Insert After\n3-Insert at End\n4-Del First\n5-Del After\n6-Del End\n7-Display Next n Prevous\n0-Exit");

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

if (opt == 1)

{

Console.WriteLine("Enter Value:");

int val1 = int.Parse(Console.ReadLine());

obj.InsertAtBeg(val1);

obj.display();

}

else if (opt == 2)

{

Console.WriteLine("Enter Old value:");

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

Console.WriteLine("Enter new value:");

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

obj.InsertAfter(oldv, newv);

obj.display();

}

else if (opt == 3)

{

Console.WriteLine("Enter val:");

int val2 = int.Parse(Console.ReadLine());

obj.InsertAtEnd(val2);

obj.display();

}

else if (opt == 4)

{

obj.DelFirst();

obj.display();

}

else if (opt == 6)

{

obj.DelEnd();

obj.display();

}

else if (opt == 5)

{

Console.WriteLine("Enter Old value : ");

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

obj.DelAfter(oldval);

obj.display();

}

else if (opt == 7)

{

Console.WriteLine("Enter val:");

int val3 = int.Parse(Console.ReadLine());

obj.PreDisplay(val3);

}

else if (opt == 0)

{

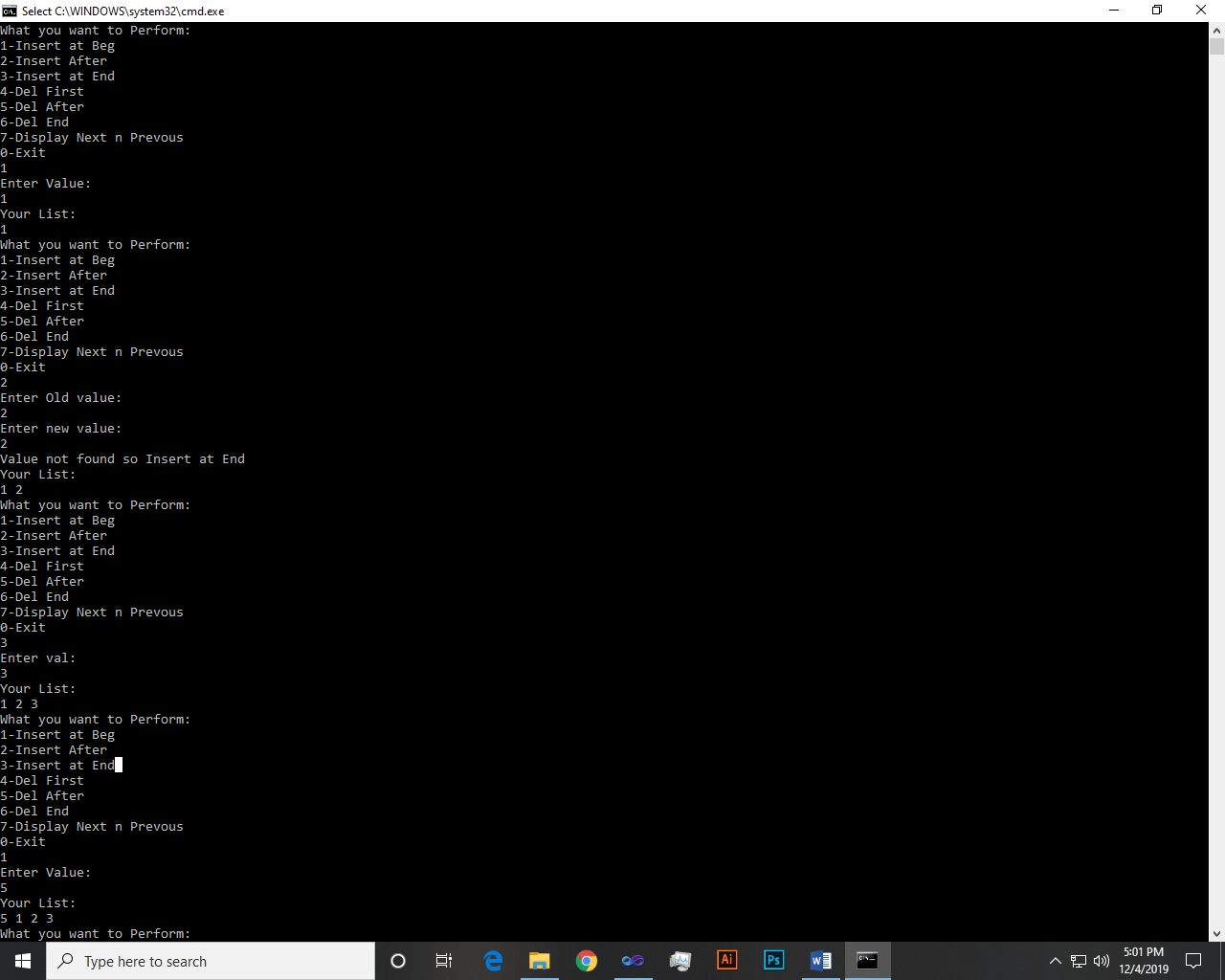
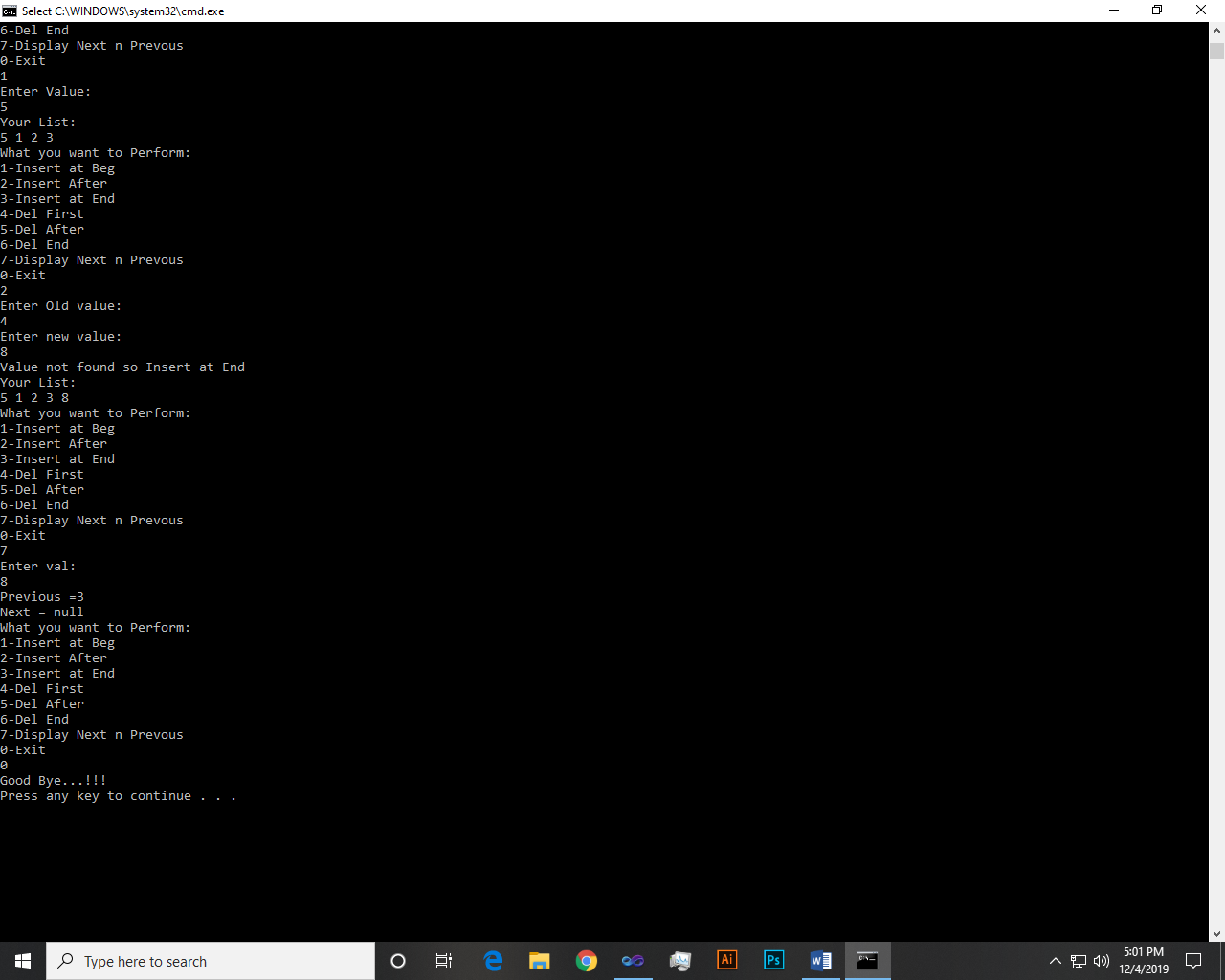
Console.WriteLine("Good Bye...!!!");

Environment.Exit(0);

}

}

Output:-

Q: - Write a program to create a circular linked list.

Input:-

public void Display()

{

int count = 1;

if (!Underflow())

{

DNode temp = start.next;

Console.WriteLine("Your list");

while (temp.next != null)

{

Console.Write(temp.data + " ");

temp = temp.next;

if(temp.next==null && count <= 1)

{

Console.Write(temp.data);

Console.Write(" \*\* ");

temp = start.next;

count++;

}

}

Console.Write(temp.data);

Console.WriteLine();

}

else

{

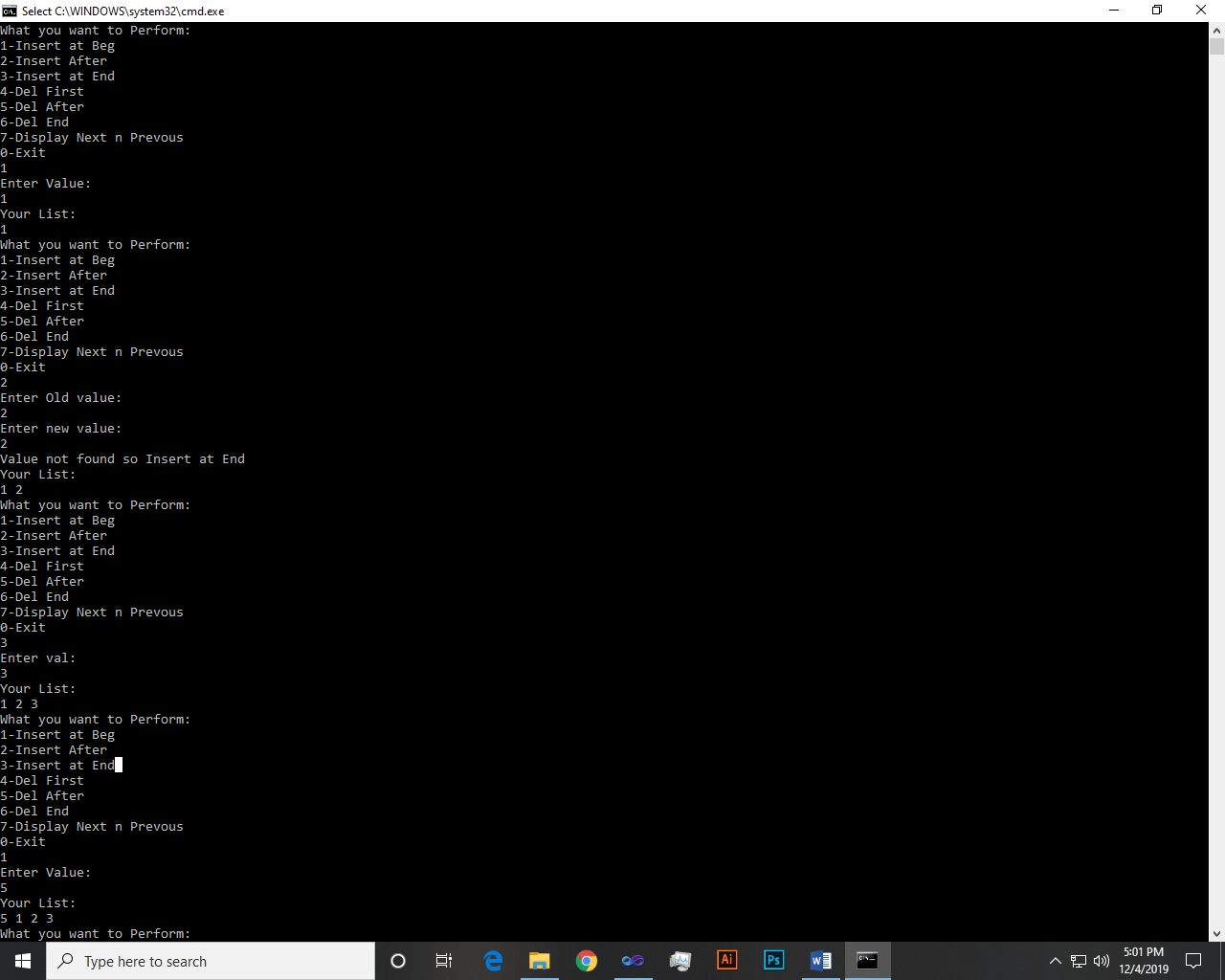
Console.WriteLine("List empty");

}

}

**Rest all methods are same as task1 and program class as well.**

Output:-



Q: - Implement Dynamic queue using double linked list.

Input:-

Class Dqueue:-

class DQueue

{

Dlinklist obj = new Dlinklist();

public bool Enqueue(int val)

{

obj.InsertAtEnd(val);

return true;

}

public bool Dequeue()

{

obj.DelFirst();

return true;

}

public void Display()

{

obj.display();

}

}

Main:-

DQueue obj = new DQueue();

while (true)

{

Console.WriteLine("What you want to Perform:");

Console.WriteLine("1-Enqueue\n2-Dequeue\n0-Exit");

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

if (opt == 1)

{

Console.WriteLine("Enter Value:");

int val1 = int.Parse(Console.ReadLine());

obj.Enqueue(val1);

obj.Display();

}

if (opt == 2)

{

obj.Dequeue();

obj.Display();

}

else if (opt == 0)

{

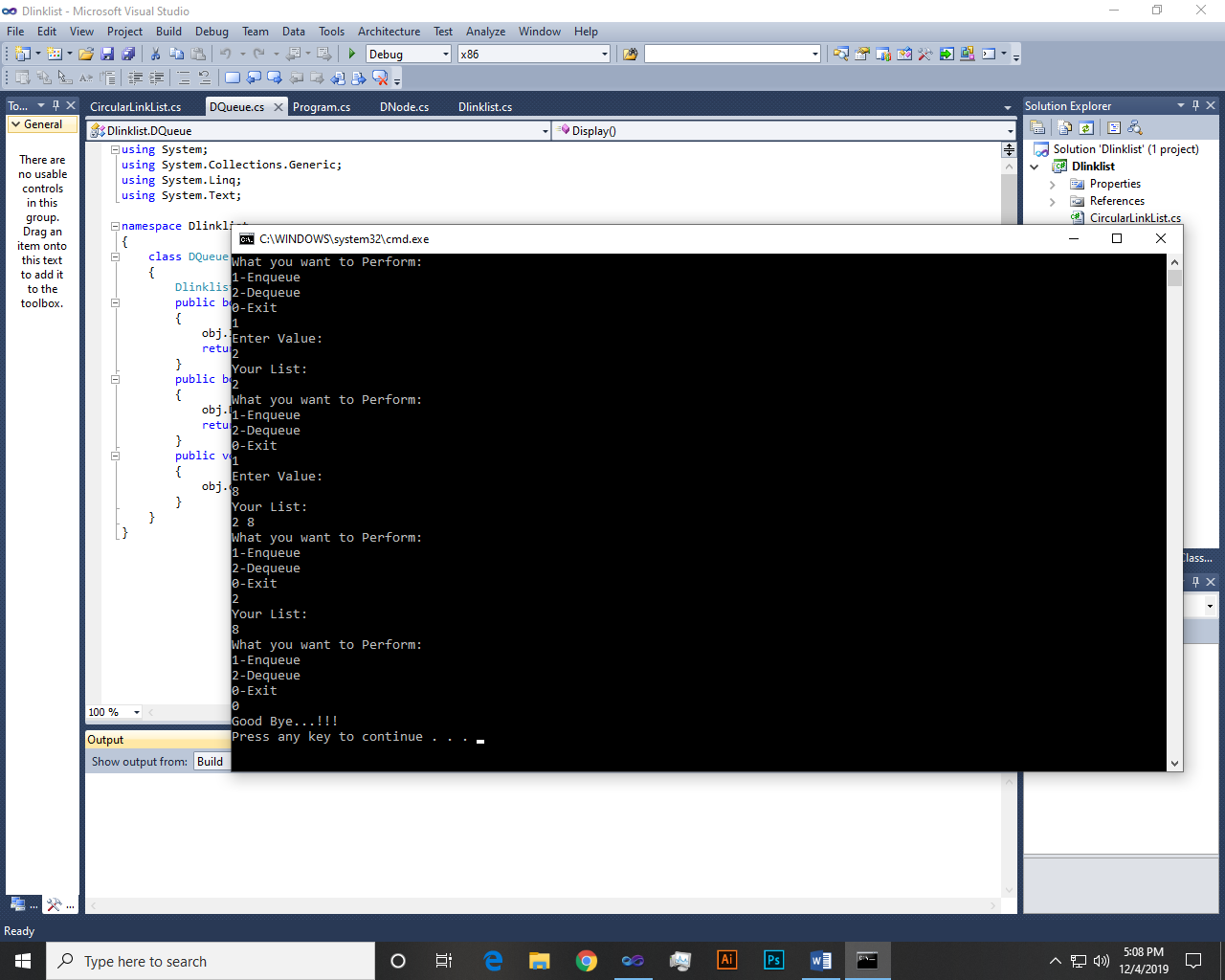
Console.WriteLine("Good Bye...!!!");

Environment.Exit(0);

}

}

Output:-



~~~~~~\*\*/**THE END**/\*\*~~~~~~