**Question 1:** Implement doubly linked list**.**

**Code:**

**Program Class:**

class Program

{

static void Main(string[] args)

{

DLinkedList obj = new DLinkedList();

while (true)

{

Console.WriteLine("What action you want to perform?");

Console.WriteLine("1-Add item in the beginning\n2-Add at last\n3-Add after any number");

Console.WriteLine("4-Remove first\n5-Remove last \n6-Remove after\n0-Exit");

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

if (choice == 1)

{

Console.WriteLine("Enter new value");

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

if (obj.InsertAtBeginning(value))

{

Console.WriteLine("Value added");

obj.Display();

}

}

else if (choice == 2)

{

Console.WriteLine("Enter new value");

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

if (obj.InsertAtEnd(value))

{

Console.WriteLine("Value added");

obj.Display();

}

}

else if (choice == 3)

{

Console.WriteLine("Enter value after which you want to add number");

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

Console.WriteLine("Enter new value");

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

if (obj.InsertAfter(value, newvalue))

{

Console.WriteLine("Value added");

obj.Display();

}

}

else if (choice == 4)

{

if (obj.RemoveFirst())

{

Console.WriteLine("Value removed");

obj.Display();

}

else

{

Console.WriteLine("an error occured");

}

}

else if (choice == 5)

{

if (obj.RemoveLast())

{

Console.WriteLine("Value removed");

obj.Display();

}

else

{

Console.WriteLine("An error occured.");

}

}

else if (choice == 6)

{

Console.WriteLine("Enter value after which you want to remove number");

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

if (obj.RemoveAfter(value))

{

Console.WriteLine("Value removed");

obj.Display();

}

else

{

Console.WriteLine("Value not found");

obj.Display();

}

}

else if (choice == 0)

{

break;

}

else

{

Console.WriteLine("Kindly select correct option");

}

}

**DLinkedList Class:**

class DLinkedList

{

DNode start;

DNode end;

public DLinkedList()

{

start = new DNode();

end = new DNode();

}

public bool Underflow()

{

if (start.next != null)

{

return false;

}

else

{

return true;

}

}

public bool InsertAtBeginning(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;

end.previous = n;

return true;

}

}

public bool InsertAtEnd(int val)

{

DNode n = new DNode(val);

if (!Underflow())

{

n.previous = end.previous;

end.previous.next = n;

end.previous = n;

return true;

}

else

{

start.next = n;

end.previous = n;

return true;

}

}

public bool InsertAfter(int val, int newval)

{

DNode n = new DNode(newval);

if (!Underflow())

{

DNode temp = start.next;

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

{

if (temp.data == val)

{

n.next = temp.next;

n.previous = temp;

temp.next.previous = n;

temp.next = n;

return true;

}

else

{

temp = temp.next;

}

}

if (InsertAtEnd(newval))

{

Console.WriteLine("Value not found");

Console.WriteLine("New value inserted at the end");

return true;

}

return false;

}

else

{

if (InsertAtEnd(newval))

{

Console.WriteLine("Value not found");

Console.WriteLine("New value inserted at the end");

return true;

}

return false;

}

}

public bool RemoveFirst()

{

if (!Underflow())

{

if (start.next.next != null)

{

start.next = start.next.next;

start.next.previous = null;

return true;

}

else

{

start.next = null;

end.previous = null;

return true;

}

}

return false;

}

public bool RemoveLast()

{

if (!Underflow())

{

if (end.previous != null)

{

if (end.previous.previous != null)

{

end.previous.previous.next = null;

end.previous = end.previous.previous;

return true;

}

else

{

end.previous = null;

start.next = null;

return true;

}

}

return false;

}

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.Write(temp.data);

Console.WriteLine();

}

else

{

Console.WriteLine("List empty");

}

}

public bool RemoveAfter(int val)

{

if (!Underflow())

{

DNode temp = start.next;

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

{

if (temp.data == val)

{

if (temp.next.next != null)

{

temp.next.next.previous = temp;

temp.next = temp.next.next;

return true;

}

else

{

if (temp.next == end.previous)

{

end.previous = temp;

}

temp.next.previous = null;

temp.next = null;

return true;

}

}

else

{

temp = temp.next;

}

}

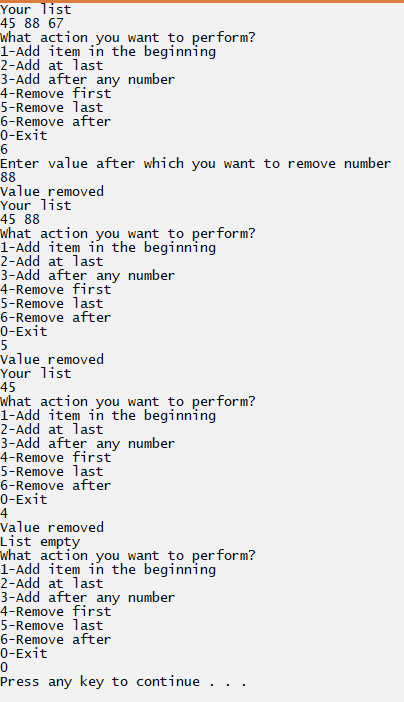
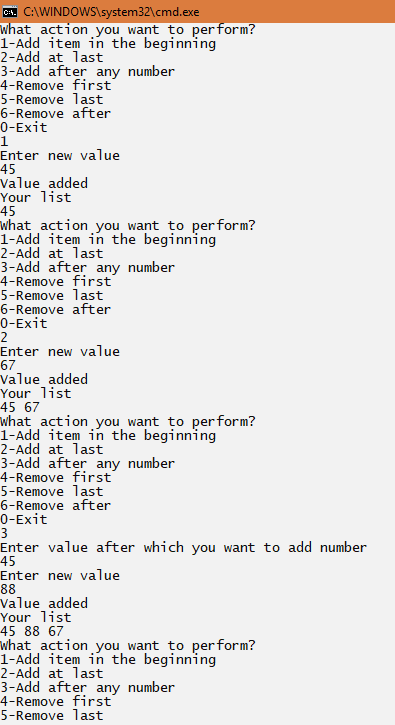
}

return false;

}

}

**Output:**



**Question 2:** Implement circular Linked list**.**

**Code:**

**DLinkedList Class:**

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:**

