**Task No. 1:**

Write a program to create a doubly linked list. Implement any sorting algorithm on the listand search for the user asked value.  
**Solution:**

class Node

{

public Node prev;

public int info;

public Node next;

public Node(int i)

{

info = i;

next = null;

prev = null;

}

}

}

class DoubleLinkedList

{

private Node start;

public DoubleLinkedList()

{

start = null;

}

public void DisplayList()

{

Node p;

if (start == null)

{

Console.WriteLine("List is Empty");

return;

}

p = start;

Console.Write("List is : ");

while (p != null)

{

Console.Write(p.info + " ");

p = p.next;

}

Console.WriteLine();

}

public void InsertInBeginning(int data)

{

Node temp = new Node(data);

temp.next = start;

start.prev = temp;

start = temp;

}

public void InsertInEmptyList(int data)

{

Node temp = new Node(data);

start = temp;

}

public void InsertAtEnd(int data)

{

Node p;

Node temp = new Node(data);

p = start;

while (p.next != null)

p = p.next;

p.next = temp;

temp.prev = p;

}

public void CreateList()

{

int i, n, data;

Console.Write("Enter the number of nodes : ");

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

if (n == 0)

return;

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

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

InsertInEmptyList(data);

for (i = 2; i <= n; i++)

{

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

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

InsertAtEnd(data);

}

}

public void InsertAfter(int data, int x)

{

Node temp = new Node(data);

Node p = start;

while (p != null)

{

if (p.info == x)

break;

p = p.next;

}

if (p == null)

{

Console.WriteLine(x + "not present in the list ");

}

else

{

temp.prev = p;

temp.next = p.next;

if (p.next != null)

p.next.prev = temp; //should not be done with p refers to last node

p.next = temp;

}

}

public void InsertBefore(int data, int x)

{

//if list is empty//

if (start == null)

{

Console.WriteLine("List is empty");

return;

}

//x is in first node, new node is to be inserted before first node //

if (start.info == x)

{

Node temp = new Node(data);

temp.next = start;

start.prev = temp;

start = temp;

return;

}

// find reference to predecessor on node conataining x

Node p = start;

while (p != null)

{

if (p.info == x)

break;

p = p.next;

}

if (p == null)

{

Console.WriteLine(x + "not present in the list");

}

else

{

Node temp = new Node(data);

temp.prev = p.prev;

temp.next = p;

p.prev.next = temp;

p.prev = temp;

}

}

public void DeleteFirstNode()

{

if (start == null) // list is empty

return;

if (start.next == null) // list has only one node

{

start = null;

return;

}

start = start.next;

start.prev = null;

}

public void DeleteLastNode()

{

if (start == null) //list is empty

return;

if (start.next == null) // list has only one node

{

start = null;

return;

}

Node p = start;

while (p.next != null)

p = p.next;

p.prev.next = null;

}

public void DeleteNode(int x)

{

if (start == null)

{

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

return;

}

if (start.next == null) // list has only one node

{

if (start.info == x)

start = null;

else

Console.WriteLine(x + "Not found");

return;

}

//Deletion of first node

if (start.info == x)

{

start = start.next;

start.prev = null;

return;

}

Node p = start.next;

while (p.next != null)

{

if (p.info == x)

break;

p = p.next;

}

if (p.next != null) //node to be delted is in between

{

p.prev.next = p.next;

p.next.prev = p.prev;

}

else // p refers to last node

{

if (p.info == x) //node to be deleted is last node

p.prev.next = null;

else

Console.WriteLine(x + "not found");

}

}

public void ReverseList()

{

if (start == null)

return;

Node p1 = start;

Node p2 = p1.next;

p1.next = null;

p1.prev = p2;

while (p2 != null)

{

p2.prev = p2.next;

p2.next = p1;

p1 = p2;

p2 = p2.prev;

}

start = p1;

}

public bool Search(int x)

{

int position = 1;

Node p = start;

while (p != null)

{

if (p.info == x)

break;

position++;

p = p.next;

}

if (p == null)

{

Console.WriteLine(x + "not found in list");

return false;

}

else

{

Console.WriteLine(x + "is at position " + position);

return true;

}

}

}

}

class Program

{

static void Main(string[] args)

{

int choice, data, x;

DoubleLinkedList list = new DoubleLinkedList();

list.CreateList();

while (true)

{

Console.WriteLine("1.Display List");

Console.WriteLine("2.Insert in empty list ");

Console.WriteLine("3.Insert a node in the Beggining of list");

Console.WriteLine("4.Insert a node at the end of list");

Console.WriteLine("5.Insert a node after a specified node");

Console.WriteLine("6.Insert a node befor a specified node");

Console.WriteLine("7.Delete first node");

Console.WriteLine("8.Delete last node");

Console.WriteLine("9.Delete any node");

Console.WriteLine("10.Reverse the list");

Console.WriteLine("11.search for an element");

Console.WriteLine("12.Quit");

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

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

if (choice == 12)

break;

switch (choice)

{

case 1:

list.DisplayList();

break;

case 2:

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

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

list.InsertInEmptyList(data);

break;

case 3:

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

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

list.InsertInBeginning(data);

break;

case 4:

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

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

list.InsertAtEnd(data);

break;

case 5:

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

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

Console.Write("Enter the element after which to insert : ");

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

list.InsertAfter(data, x);

break;

case 6:

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

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

Console.Write("Enter the element before which to insert : ");

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

list.InsertBefore(data, x);

break;

case 7:

list.DeleteFirstNode();

break;

case 8:

list.DeleteLastNode();

break;

case 9:

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

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

list.DeleteNode(data);

break;

case 10:

list.ReverseList();

break;

case 11:

Console.WriteLine("Enter the element to be searched : ");

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

list.Search(data);

break;

default:

Console.WriteLine("Wrong choice");

break;

}

Console.WriteLine();

}

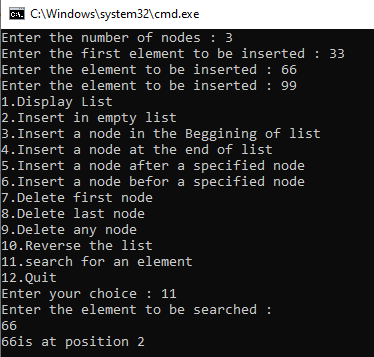
Console.WriteLine("Exiting");

}

}

}

**Output:**



**Task No. 2:**

Write a program to create a circular linked list.

**Solution:**

class Node

{

public Node link;

public int info;

public Node(int i)

{

info = i;

link = null;

}

}

}

class CircularLinkedList

{

private Node last;

public CircularLinkedList()

{

last = null;

}

public void DisplayList()

{

Node p;

if (last == null)

{

Console.WriteLine("List is Empty\n");

return;

}

Console.Write("List is : ");

p = last.link;

do

{

Console.Write(p.info + " ");

p = p.link;

}

while (p != last.link);

Console.WriteLine();

}

public void InsertInBeginning(int data)

{

Node temp = new Node(data);

temp.link = last.link;

last.link = temp;

}

public void InsertInEmptyList(int data)

{

Node temp = new Node(data);

last = temp;

last.link = last;

}

public void InsertAtEnd(int data)

{

Node temp = new Node(data);

temp.link = last.link;

last.link = temp;

last = temp;

}

public void CreateList()

{

int i, n, data;

Console.Write("Enter the number of nodes : ");

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

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

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

InsertInEmptyList(data);

if (n == 0)

return;

for (i = 2; i <= n; i++)

{

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

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

InsertAtEnd(data);

}

}

public void InsertAfter(int data, int x)

{

Node p = last.link;

do

{

if (p.info == x)

break;

p = p.link;

} while (p != last.link);

if (p == last.link && p.info != x)

{

Console.WriteLine(x + " not present in the list : ");

}

else

{

Node temp = new Node(data);

temp.link = p.link;

p.link = temp;

if (p == last)

last = temp; ;

}

}

public void DeleteFirstNode()

{

if (last == null) //list is empty

return;

if (last.link == last) //list has only one node

{

last =null;

return;

}

last.link = last.link.link;

}

public void DeleteLastNode()

{

if (last == null) //list is empty

return;

if (last.link == last) //list has only one node

{

last = null;

return;

}

Node p = last.link;

while (p.link != last)

p = p.link;

p.link = last.link;

last = p;

}

public void DeleteNode(int x)

{

if (last == null) //list is empty

return;

if (last.link == last && last.info ==x) //deletion of only node

{

last = null;

return;

}

if (last.link.info == x ) //deletion of first node

{

last.link = last.link.link;

return;

}

// Deletion in between or at the end

Node p = last.link;

while (p.link != last.link)

{

if (p.link.info == x)

break;

p = p.link;

}

if (p.link == last.link)

Console.WriteLine("Element" + x + " not in the list");

else

{

p.link = p.link.link;

if (last.info == x)

last = p;

}

}

}

}

class Program

{

static void Main(string[] args)

{

int choice, data, x;

CircularLinkedList list = new CircularLinkedList();

list.CreateList();

while (true)

{

Console.WriteLine("1.Display List");

Console.WriteLine("2.Insert in empty list ");

Console.WriteLine("3.Insert a node in the Beggining");

Console.WriteLine("4.Insert a node at the end of list");

Console.WriteLine("5.Insert a node after a specified node");

Console.WriteLine("6.Delete first node");

Console.WriteLine("7.Delete last node");

Console.WriteLine("8.Delete any node");

Console.WriteLine("9.Quit");

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

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

if (choice == 9)

break;

switch (choice)

{

case 1:

list.DisplayList();

break;

case 2:

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

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

list.InsertInEmptyList(data);

break;

case 3:

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

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

list.InsertInBeginning(data);

break;

case 4:

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

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

list.InsertAtEnd(data);

break;

case 5:

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

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

Console.Write("Enter the element after which to insert : ");

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

list.InsertAfter(data, x);

break;

case 6:

list.DeleteFirstNode();

break;

case 7:

list.DeleteLastNode();

break;

case 8:

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

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

list.DeleteNode(data);

break;

default:

Console.WriteLine("Wrong choice");

break;

}

Console.WriteLine();

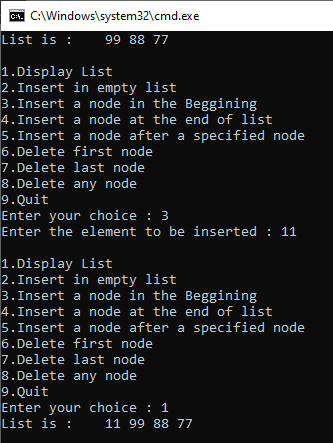
}

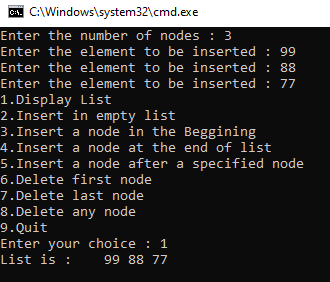
Console.WriteLine("Exiting");

}

}

}

**Output:**



**Task No. 3:**

Implement Dynamic queue using double linked list.

**Solution:**

class Node

{

public Node prev;

public int info;

public Node next;

public Node(int i)

{

info = i;

next = null;

prev = null;

}

}

}

class DynamicQueue

{

private Node start;

public DynamicQueue()

{

start = null;

}

public void DisplayQueue()

{

Node p;

if (start == null)

{

Console.WriteLine("Queue is Empty");

return;

}

p = start;

Console.Write("Queue is : ");

while (p != null)

{

Console.Write(p.info + " ");

p = p.next;

}

Console.WriteLine();

}

public void InsertInEmptyQueue(int data)

{

Node temp = new Node(data);

start = temp;

}

public void InsertElement(int data)

{

Node p;

Node temp = new Node(data);

p = start;

while (p.next != null)

p = p.next;

p.next = temp;

temp.prev = p;

}

public void CreateQueue()

{

int i, n, data;

Console.Write("Enter the number of elements in Queue : ");

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

if (n == 0)

return;

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

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

InsertInEmptyQueue(data);

for (i = 2; i <= n; i++)

{

Console.Write("Enter next element to be inserted in Queue : ");

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

InsertElement(data);

}

}

public void DeleteElement()

{

if (start == null) // Queue is empty

return;

if (start.next == null) // Queue has only one element

{

start = null;

return;

}

start = start.next;

start.prev = null;

}

public bool SearchElement(int x)

{

int position = 1;

Node p = start;

while (p != null)

{

if (p.info == x)

break;

position++;

p = p.next;

}

if (p == null)

{

Console.WriteLine(x + "not found in list");

return false;

}

else

{

Console.WriteLine(x + "is at position " + position);

return true;

}

}

}

}

class Program

{

static void Main(string[] args)

{

int choice, data;

DynamicQueue Queue = new DynamicQueue();

Queue.CreateQueue();

while (true)

{

Console.WriteLine("1.Display Queue");

Console.WriteLine("2.Insert in empty Queue ");

Console.WriteLine("3.Insert an element in the Queue");

Console.WriteLine("4.Delete an element from the Queue");

Console.WriteLine("5.search for an element in the Queue");

Console.WriteLine("6.Quit");

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

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

if (choice == 6)

break;

switch (choice)

{

case 1:

Queue.DisplayQueue();

break;

case 2:

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

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

Queue.InsertInEmptyQueue(data);

break;

case 3:

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

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

Queue.InsertElement(data);

break;

case 4:

Queue.DeleteElement();

break;

case 5:

Console.WriteLine("Enter the element to be searched : ");

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

Queue.SearchElement(data);

break;

default:

Console.WriteLine("Wrong choice");

break;

}

Console.WriteLine();

}

Console.WriteLine("Exiting");

}

}

}

**Output:**

