**Task No. 1:** Write a program to create a linked list.

**Solution:**

class Node

{

public int info;

public Node link;

public Node(int i)

{

info = i;

link = null;

}

}

}

class SingleLinkedList

{

private Node start;

public SingleLinkedList()

{

start = null;

}

public void DisplayList()

{

Node p;

if (start == null)

{

Console.WriteLine("List is Empty");

return;

}

Console.Write("List is : ");

p = start;

while (p != null)

{

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

p = p.link;

}

Console.WriteLine();

}

public void CountNodes()

{

int n = 0;

Node p = start;

while (p != null)

{

n++;

p = p.link;

}

Console.WriteLine("Number of nodes in the list = " + n);

}

public bool Search(int x)

{

int position = 1;

Node p = start;

while (p != null)

{

if (p.info == x)

break;

position++;

p = p.link;

}

if (p == null)

{

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

return false;

}

else

{

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

return true;

}

}

public void InsertInBeginning(int data)

{

Node temp = new Node(data);

temp.link = start;

start = temp;

}

public void InsertAtEnd(int data)

{

Node p;

Node temp = new Node(data);

if (start == null)

{

start = temp;

return;

}

p = start;

while (p.link != null)

p = p.link;

p.link = temp;

}

public void CreateList()

{

int i, n, data;

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

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

if (n == 0)

return;

for (i = 1; 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 = start;

while (p != null)

{

if (p.info == x)

break;

p = p.link;

}

if (p == null)

{

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

}

else

{

Node temp = new Node(data);

temp.link = p.link;

p.link = temp;

}

}

public void InsertBefore(int data, int x)

{

Node temp;

//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 (x == start.info)

{

temp = new Node(data);

temp.link = start;

start = temp;

return;

}

// find reference to predecessor on node conataining x

Node p = start;

while (p.link != null)

{

if (p.link.info == x)

break;

p = p.link;

}

if (p.link == null)

{

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

}

else

{

temp = new Node(data);

temp.link = p.link;

p.link = temp;

}

}

public void InsertAtPosition(int data, int k)

{

Node temp;

int i;

if (k == 1)

{

temp = new Node(data);

temp.link = start;

start = temp;

return;

}

Node p = start;

for (i = 1; i < k - 1 && p != null; i++) //find a reference to k-1 node

p = p.link;

if (p == null)

Console.WriteLine("You can only insert upto" + i + "th position");

else

{

temp = new Node(data);

temp.link = p.link;

p.link = temp;

}

}

public void DeleteFirstNode()

{

if (start == null)

return;

start = start.link;

}

public void DeleteLastNode()

{

if (start == null)

return;

if (start.link == null)

{

start = null;

return;

}

Node p = start;

while (p.link.link != null)

p = p.link;

p.link = null;

}

public void DeleteNode(int x)

{

if (start == null)

{

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

return;

}

//Deletion of first node

if (start.info == x)

{

start = start.link;

return;

}

// Deletion in between or at the end

Node p = start;

while (p.link != null)

{

if (p.link.info == x)

break;

p = p.link;

}

if (p.link == null)

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

else

p.link = p.link.link;

}

public void ReverseList()

{

Node prev, p, next;

prev = null;

p = start;

while (p != null)

{

next = p.link;

p.link = prev;

prev = p;

p = next;

}

start = prev;

}

}

}

class Program

{

static void Main(string[] args)

{

int choice, data, k, x;

SingleLinkedList list = new SingleLinkedList();

list.CreateList();

while (true)

{

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

Console.WriteLine("2.Count the number of nodes");

Console.WriteLine("3.search for an element");

Console.WriteLine("4.Insert in empty list/ Insert in beginning of list");

Console.WriteLine("5.Insert a node at the end of list");

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

Console.WriteLine("7.Insert a node befor a specified node");

Console.WriteLine("8.Insert a node at a given position");

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

Console.WriteLine("10.Delete last node");

Console.WriteLine("11.Delete any node");

Console.WriteLine("12.Reverse the list");

Console.WriteLine("13.Quit");

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

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

if (choice == 13)

break;

switch (choice)

{

case 1:

list.DisplayList();

break;

case 2:

list.CountNodes();

break;

case 3:

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

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

list.Search(data);

break;

case 4:

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

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

list.InsertInBeginning(data);

break;

case 5:

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

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

list.InsertAtEnd(data);

break;

case 6:

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 7:

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 8:

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

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

Console.Write("Enter the position at which to insert : ");

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

list.InsertBefore(data, k);

break;

case 9:

list.DeleteFirstNode();

break;

case 10:

list.DeleteLastNode();

break;

case 11:

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

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

list.DeleteNode(data);

break;

case 12:

list.ReverseList();

break;

default:

Console.WriteLine("Wrong choice");

break;

}

Console.WriteLine();

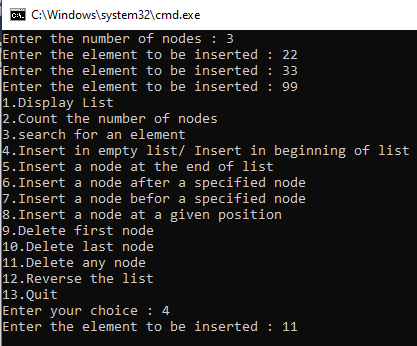
}

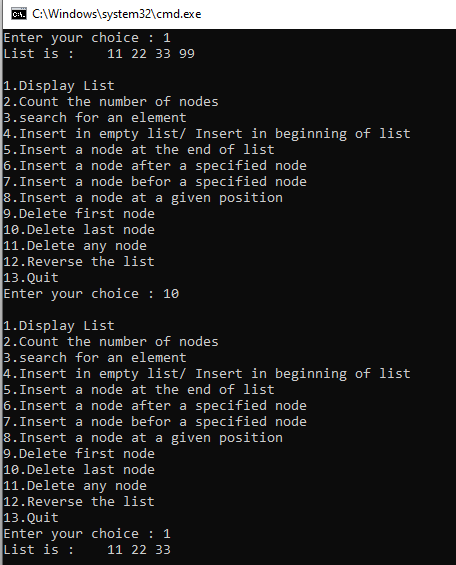
Console.WriteLine("Exiting....");

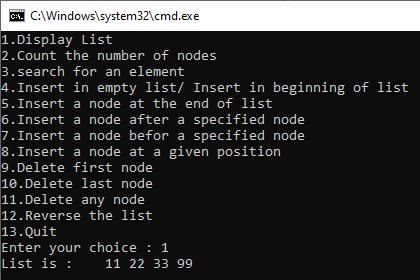
}

}

}

**Output:**





**Task No. 2:** Write a program to implement dynamic stacks by using linked list.

**Solution:**

class Node

{

public int info;

public Node link;

public Node(int i)

{

info = i;

link = null;

}

}

}

class DynamicStack

{

private Node start;

public DynamicStack()

{

start = null;

}

public void DisplayStackList()

{

Node p;

if (start == null)

{

Console.WriteLine("Stack is Empty");

return;

}

Console.Write("Stack elements are : ");

p = start;

while (p != null)

{

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

p = p.link;

}

Console.WriteLine();

}

public void CountElements()

{

int n = 0;

Node p = start;

while (p != null)

{

n++;

p = p.link;

}

Console.WriteLine("Number of elements in the stack = " + n);

}

public bool Search(int x)

{

int position = 1;

Node p = start;

while (p != null)

{

if (p.info == x)

break;

position++;

p = p.link;

}

if (p == null)

{

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

return false;

}

else

{

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

return true;

}

}

public void InsertElement(int data)

{

Node p;

Node temp = new Node(data);

if (start == null)

{

start = temp;

return;

}

p = start;

while (p.link != null)

p = p.link;

p.link = temp;

}

public void CreateList()

{

int i, n, data;

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

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

if (n == 0)

return;

for (i = 1; i <= n; i++)

{

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

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

InsertElement(data);

}

}

public void DeleteElement()

{

if (start == null)

return;

if (start.link == null)

{

start = null;

return;

}

Node p = start;

while (p.link.link != null)

p = p.link;

p.link = null;

}

}

}

class Program

{

static void Main(string[] args)

{

int choice, data;

DynamicStack stackQQ = new DynamicStack();

stackQQ.CreateList();

while (true)

{

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

Console.WriteLine("2.Count the number of element in stack");

Console.WriteLine("3.search for an element");

Console.WriteLine("4.Insert an element in the stack");

Console.WriteLine("5.Delete element from stack");

Console.WriteLine("6.Quit");

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

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

if (choice == 6)

break;

switch (choice)

{

case 1:

stackQQ.DisplayStackList();

break;

case 2:

stackQQ.CountElements();

break;

case 3:

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

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

stackQQ.Search(data);

break;

case 4:

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

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

stackQQ.InsertElement(data);

break;

case 5:

stackQQ.DeleteElement();

break;

default:

Console.WriteLine("Wrong choice");

break;

}

Console.WriteLine();

}

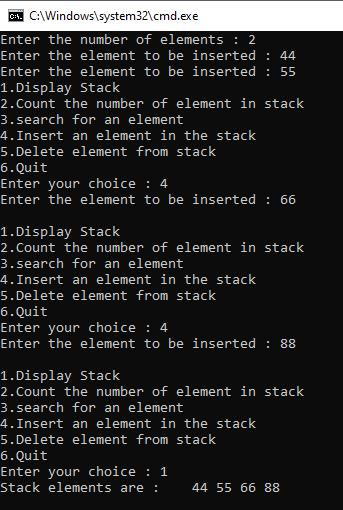
Console.WriteLine("Exiting....");

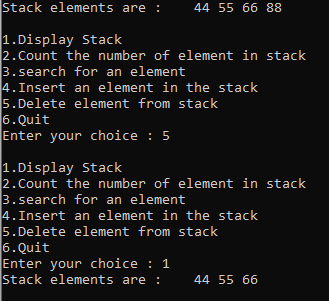
}

}

}

**Output:**





**Task No. 3:** Write a program to implement dynamic queue by using linked list.

**Solution:**

class Node

{

public int info;

public Node link;

public Node(int i)

{

info = i;

link = null;

}

}

}

class DynamicQueue

{

private Node start;

public DynamicQueue()

{

start = null;

}

public void DisplayQueue()

{

Node p;

if (start == null)

{

Console.WriteLine("Queue is Empty");

return;

}

Console.Write("Queue is : ");

p = start;

while (p != null)

{

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

p = p.link;

}

Console.WriteLine();

}

public void CountElements()

{

int n = 0;

Node p = start;

while (p != null)

{

n++;

p = p.link;

}

Console.WriteLine("Number of elements in the Queue = " + n);

}

public bool SearchElement(int x)

{

int position = 1;

Node p = start;

while (p != null)

{

if (p.info == x)

break;

position++;

p = p.link;

}

if (p == null)

{

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

return false;

}

else

{

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

return true;

}

}

public void InsertElement(int data)

{

Node p;

Node temp = new Node(data);

if (start == null)

{

start = temp;

return;

}

p = start;

while (p.link != null)

p = p.link;

p.link = temp;

}

public void CreateQueue()

{

int i, n, data;

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

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

if (n == 0)

return;

for (i = 1; i <= n; i++)

{

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

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

InsertElement(data);

}

}

public void DeleteElement()

{

if (start == null)

return;

start = start.link;

}

}

}

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.Count the number of elements in Queue");

Console.WriteLine("3.search for an element in Queue");

Console.WriteLine("4.Insert an element in Queue");

Console.WriteLine("5.Delete element from 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:

Queue.CountElements();

break;

case 3:

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

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

Queue.SearchElement(data);

break;

case 4:

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

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

Queue.InsertElement(data);

break;

case 5:

Queue.DeleteElement();

break;

default:

Console.WriteLine("Wrong choice");

break;

}

Console.WriteLine();

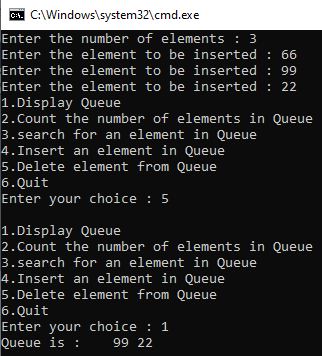
}

Console.WriteLine("Exiting....");

}

}

}

**Output:**

