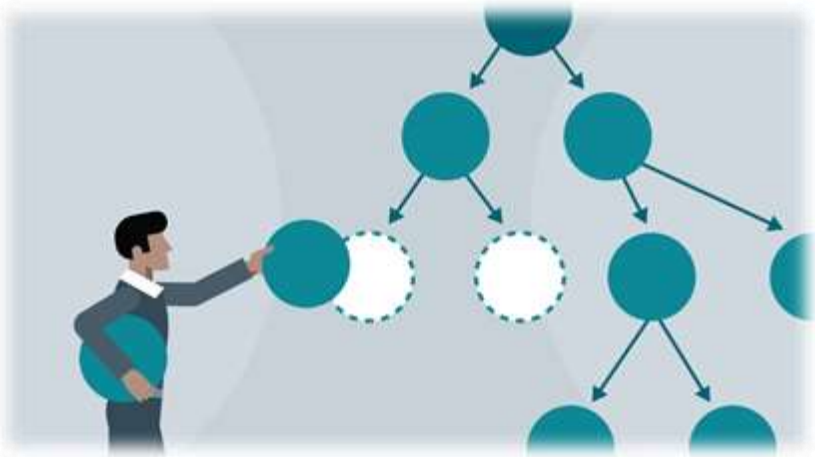
Data Structure

&

Algorithms



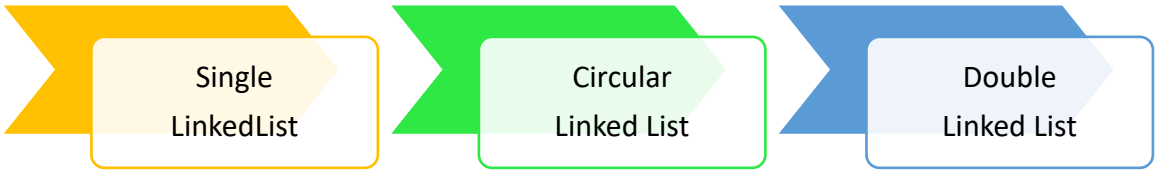
M.Rizwan

Computer Lecturer

Full Stack .NET Developer/Trainer

Linked-List

Implementation using **Visual C++** & **Visual C#**



Single Linked-List

Implementation using Visual C++ & Visual C#

SLL1 – Insertion at the Beginning of a Linked List (Visual C++ Implementation)

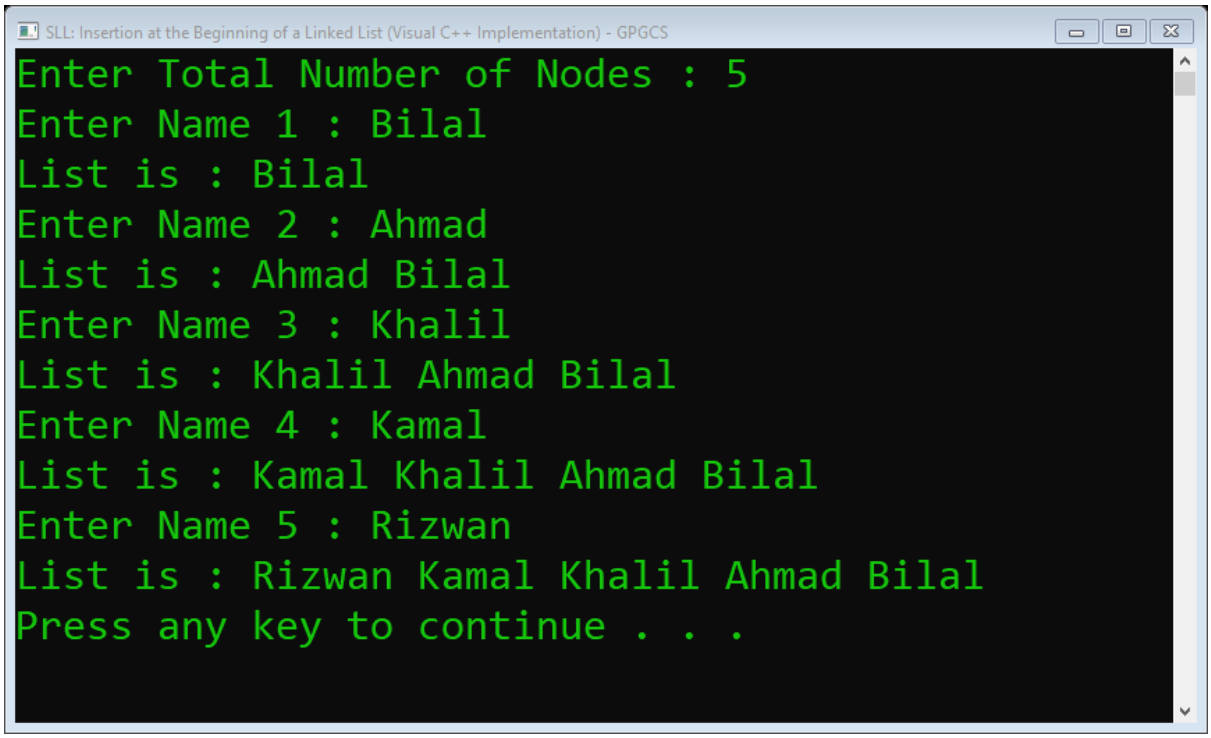
SinglyLinkedList.cpp

```
#include <iostream>
using namespace std;

namespace LinkedList
{
	struct Node
	{
		char name[20];
		Node *link;
	};

	class SLL
	{
		private:
			Node * start, *temp;
		public:
			SLL() {
				start = temp = NULL;
			}
			// Insertion in SLL (AddFirst)
			void InsertNodes(char _name[]) {
				temp = new Node;
				strcpy_s(temp->name, _name);
				temp->link = start;
				start = temp;
			}
			// Traversing in SLL
			void DisplayNodes() {
				temp = start;
				cout << "List is : ";
				while (temp != NULL)
				{
					cout << temp->name << " ";
					temp = temp->link;
				}
				cout << endl;
			}
	};
}
```

Main.cpp

```
#include <iostream>
#include "SinglyLinkedList.cpp"
using namespace std;
using namespace LinkedList;

int main() {

	SLL obj;

	char name[20];
	int n;
	cout << "Enter Total Number of Nodes : ";
	cin >> n;
	for (int i = 0; i < n; i++)
	{
		cout << "Enter Name " << i + 1 << " : ";
		cin >> name;
		obj.InsertNodes(name);
		obj.DisplayNodes();
	}

	system("pause");
	return 0;
}
```

Output



SLL1 – Insertion at the Beginning of a Linked List (Visual C# Implementation)

SLL.cs

```
using System;
using static System.Console;

namespace LinkedList
{
    public class Node
    {
        public object data;
        public Node link;
    }
    public class SLL
    {
        private Node start;
        private Node temp;
        public SLL()
        {
            start = null;
        }
        // Insertion in SLL (AddFirst)
        public void InsertNodes(object _data)
        {
            temp = new Node();
            temp.data = _data;
            temp.link = start;
            start = temp;
        }
        // Traversing in SLL
        public void DisplayNodes()
        {
            temp = start;
            Write("List is : ");
            while (temp != null)
            {
                Write($"{temp.data} ");
                temp = temp.link;
            }
            WriteLine();
        }
    }
}
```

Program.cs

```
using System;
using static System.Console;
using static System.Convert;
using LinkedList;
```

```
namespace SLL3_CSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            SLL obj = new SLL();
            Write("Enter Total Number of Nodes : ");
            int n = ToInt32(ReadLine());
            for (int i = 0; i < n; i++)
            {
                Write($"Enter Value {i + 1} : ");
                object value = ReadLine();
                obj.InsertNodes(value);
                obj.DisplayNodes();
            }

            ReadKey(true);
        }
    }
}
```

SLL2 – Insertion at the End of a Linked List (Visual C++ Implementation)

SinglyLinkedList.cpp

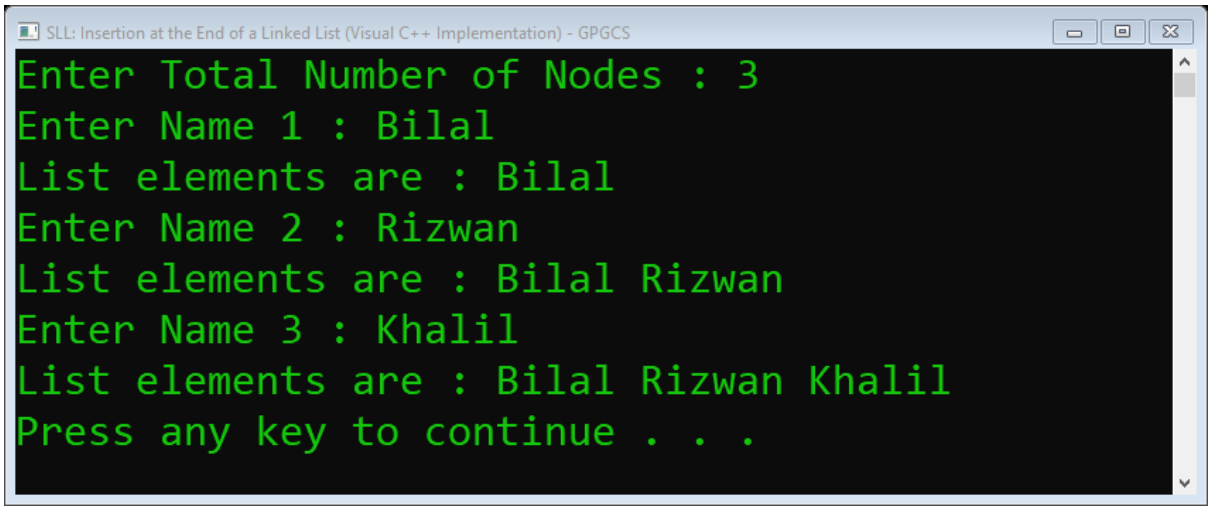
```
#include <iostream>
#include <string>
using namespace std;

namespace LinkedList
{
	struct Node
	{
		string name;
		Node *link;
	};

	class SLL
	{
		private:
			Node * start, *temp, *current;
		public:
			SLL() {
				start = NULL;
				temp = NULL;
				current = NULL;
			}
			// Insertion in SLL (AddLast)
			void InsertNodesAtEnd(string _name) {
				temp = new Node;
				temp->name = _name;
				temp->link = NULL;
```

```
                        if (start == NULL)
                                start = temp;
                        else
                        {
                                current = start;
                                while (current->link != NULL)
                                        current = current->link;
                                current->link = temp;
                        }
                }
                // Traversing in SLL
                void DisplayNodes() {
                        current = start;
                        cout << "List elements are : ";
                        while (current != NULL)
                        {
                                cout << current->name << " ";
                                current = current->link;
                        }
                        cout << endl;
                }
        };
}
```

Main.cpp

```
#include <iostream>
#include "SinglyLinkedList.cpp"
using namespace std;
using namespace LinkedList;

int main() {

        SLL obj;

        string name;
        int length;
        cout << "Enter Total Number of Nodes : ";
        cin >> length;
        for (int i = 0; i < length; i++)
        {
                cout << "Enter Name " << i + 1 << " : ";
                cin >> name;
                obj.InsertNodesAtEnd(name);
                obj.DisplayNodes();
        }

        system("pause");
        return 0;
}
```

Output



SLL3 – Insertion at a Specified Location within a Linked List

(Visual C++ Implementation)

SinglyLinkedList.cpp

```
#include <iostream>
#include <string>
using namespace std;

namespace LinkedList
{
	struct Node
	{
		string name;
		Node *link;
	};

	class SLL
	{
		private:
			Node * start, *temp, *current;
		public:
			SLL() {
				start = NULL;
				temp = NULL;
				current = NULL;
			}
			// Create a Singly-Linked-List
			void AddNodes(string _name) {
				temp = new Node;
				temp->name = _name;
				temp->link = NULL;
				if (start == NULL)
					start = temp;
```

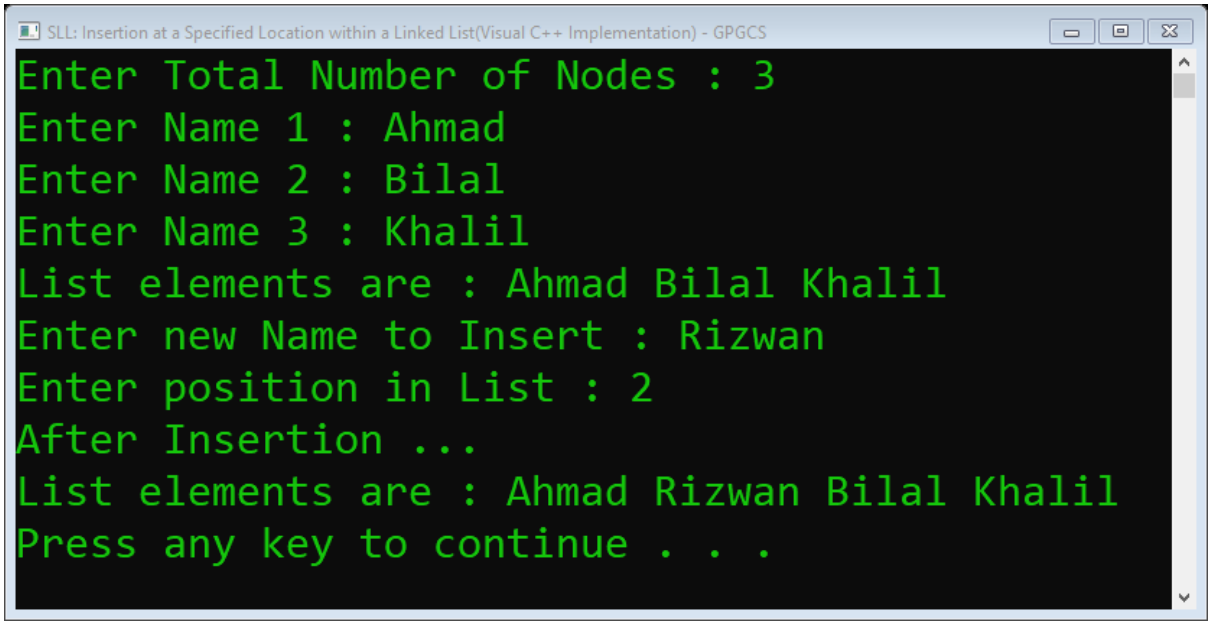
```
                        else
                        {
                                current = start;
                                while (current->link != NULL)
                                        current = current->link;
                                current->link = temp;
                        }
                }
                // Insertion at a Specified Location in SLL
                void InsertNode(string _name, int pos) {
                        temp = new Node;
                        temp->name = _name;
                        // Go to the specified position
                        current = start;
                        for (int i = 1; i < pos-1; i++){
                                current = current->link;
                                if (current == NULL) {
                                        cout << "Invalid Position";
                                        return;
                                }
                        }
                        // Insert Node in SLL
                        if (pos == 1) {
                                temp->link = start;
                                start = temp;
                                return;
                        }
                        temp->link = current->link;
                        current->link = temp;
                }
                // Traversing in SLL
                void DisplayNodes() {
                        current = start;
                        cout << "List elements are : ";
                        while (current != NULL)
                        {
                                cout << current->name << " ";
                                current = current->link;
                        }
                        cout << endl;
                }
        };
}
```

Main.cpp

```
#include <iostream>
#include "SinglyLinkedList.cpp"
using namespace std;
using namespace LinkedList;

int main() {

        SLL obj;

        string name;
        int length;
        cout << "Enter Total Number of Nodes : ";
        cin >> length;
```

```
        for (int i = 0; i < length; i++)
        {
                cout << "Enter Name " << i + 1 << " : ";
                cin >> name;
                obj.AddNodes(name);
        }
        obj.DisplayNodes();
        cout << "Enter new Name to Insert : ";
        cin >> name;
        int position;
        cout << "Enter position in List : ";
        cin >> position;
        obj.InsertNode(name, position);
        cout << "After Insertion ..." << endl;
        obj.DisplayNodes();



        system("pause");
        return 0;
}
```

Output



SLL4 – Delete a Node at First within a Linked List (Visual C++ Implementation)

SinglyLinkedList.cpp

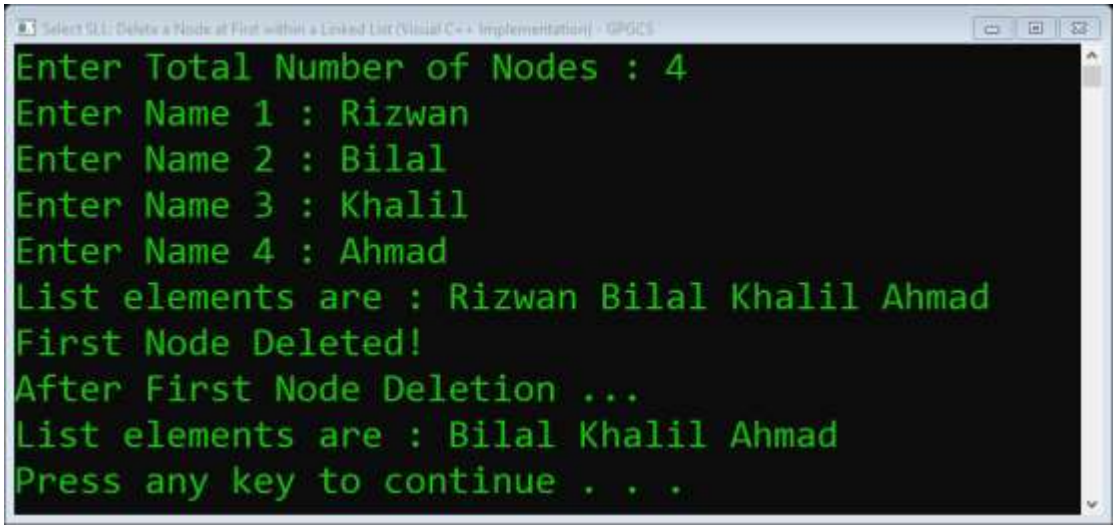
```
#include <iostream>
#include <string>
using namespace std;

namespace LinkedList
{
	struct Node
	{
		string name;
		Node *link;
	};

	class SLL
	{
		private:
			Node * start, *temp, *current;
		public:
			SLL() {
				start = NULL;
				temp = NULL;
				current = NULL;
			}
			// Create a Singly-Linked-List
			void AddNodes(string _name) {
				temp = new Node;
				temp->name = _name;
				temp->link = NULL;
				if (start == NULL)
					start = temp;
				else
				{
					current = start;
					while (current->link != NULL)
						current = current->link;
					current->link = temp;
				}
			}
			// Deletion in SLL
			void DeleteNodeAtFirst() {
				if (start == NULL)
					cout << "List is Empty!" << endl;
				else
				{
					temp = start;
					start = start->link;
					delete temp;
					cout << "First Node Deleted!" << endl;
				}
			}
			// Traversing in SLL
			void DisplayNodes() {
```

```
                        current = start;
                        cout << "List elements are : ";
                        while (current != NULL)
                        {
                                cout << current->name << " ";
                                current = current->link;
                        }
                        cout << endl;
                }
        };
}
```

Main.cpp

```
#include <iostream>
#include "SinglyLinkedList.cpp"
using namespace std;
using namespace LinkedList;

int main() {

        SLL obj;
        string name;
        int length;
        cout << "Enter Total Number of Nodes : ";
        cin >> length;
        for (int i = 0; i < length; i++)
        {
                cout << "Enter Name " << i + 1 << " : ";
                cin >> name;
                obj.AddNodes(name);
        }
        obj.DisplayNodes();
        obj.DeleteNodeAtFirst();
        cout << "After First Node Deletion ..." << endl;
        obj.DisplayNodes();

        system("pause");
        return 0;
}
```

Output



SLL5 – Delete a Node at Last within a Linked List (Visual C++ Implementation)

SinglyLinkedList.cpp

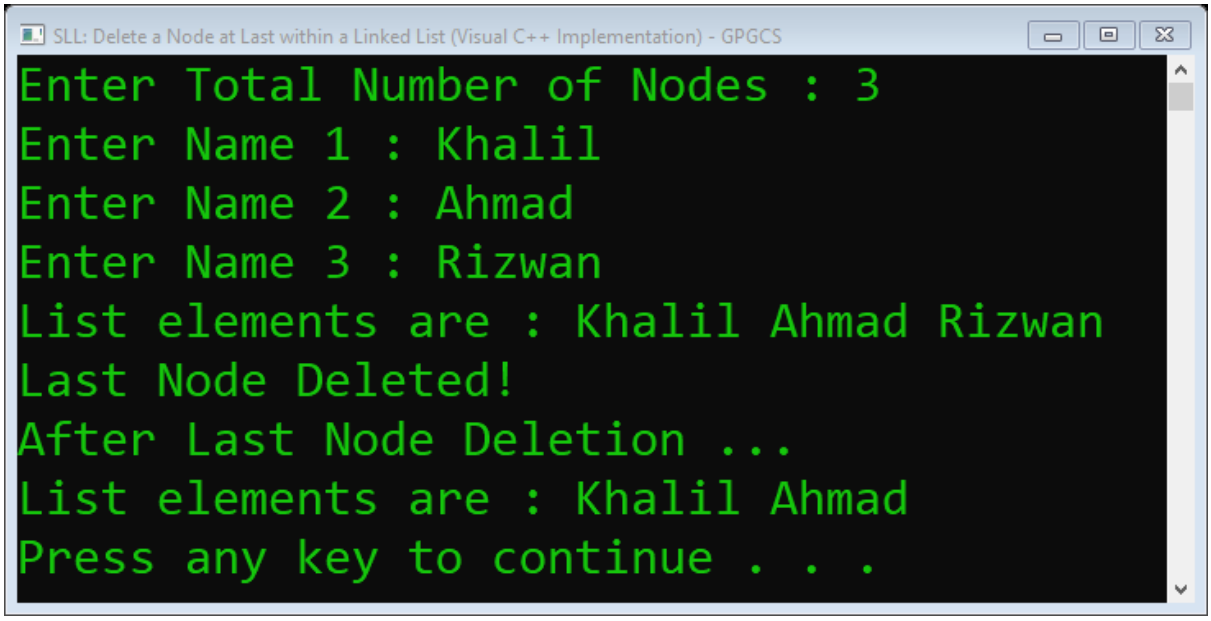
```
#include <iostream>
#include <string>
using namespace std;

namespace LinkedList
{
	struct Node
	{
		string name;
		Node *link;
	};

	class SLL
	{
		private:
			Node * start, *temp, *current;
		public:
			SLL() {
				start = NULL;
				temp = NULL;
				current = NULL;
			}
			// Create a Singly-Linked-List
			void AddNodes(string _name) {
				temp = new Node;
				temp->name = _name;
				temp->link = NULL;
				if (start == NULL)
					start = temp;
				else
				{
					current = start;
					while (current->link != NULL)
						current = current->link;
					current->link = temp;
				}
			}
			// Deletion in SLL
			void DeleteNodeAtLast() {
				if (start == NULL)
					cout << "List is Empty!" << endl;
				else
				{
					current = temp = start;
					while (current->link != NULL)
					{
						temp = current;
						current = current->link;
					}
					temp->link = NULL;
```

```
                        delete current;
                        cout << "Last Node Deleted!" << endl;
                }
        }
        // Traversing in SLL
        void DisplayNodes() {
                current = start;
                cout << "List elements are : ";
                while (current != NULL)
                {
                        cout << current->name << " ";
                        current = current->link;
                }
                cout << endl;
        }
    };
}
```

Main.cpp

```
#include <iostream>
#include "SinglyLinkedList.cpp"
using namespace std;
using namespace LinkedList;

int main() {

    SLL obj;

    string name;
    int length;
    cout << "Enter Total Number of Nodes : ";
    cin >> length;
    for (int i = 0; i < length; i++)
    {
        cout << "Enter Name " << i + 1 << " : ";
        cin >> name;
        obj.AddNodes(name);
    }
    obj.DisplayNodes();
    obj.DeleteNodeAtLast();
    cout << "After Last Node Deletion ..." << endl;
    obj.DisplayNodes();

    system("pause");
    return 0;
}
```

Output



SLL6 – Searching at a specified location within a Linked List

(Visual C++ Implementation)

SinglyLinkedList.cpp

```
#include <iostream>
#include <string>
using namespace std;

namespace LinkedList
{
	struct Node
	{
		string name;
		Node *link;
	};

	class SLL
	{
		private:
			Node * start, *temp, *current;
		public:
			SLL() {
				start = NULL;
				temp = NULL;
				current = NULL;
			}
			// Insertion in SLL
			void InsertNodes(string _name) {
				temp = new Node;
				temp->name = _name;
				temp->link = NULL;
				if (start == NULL)
```

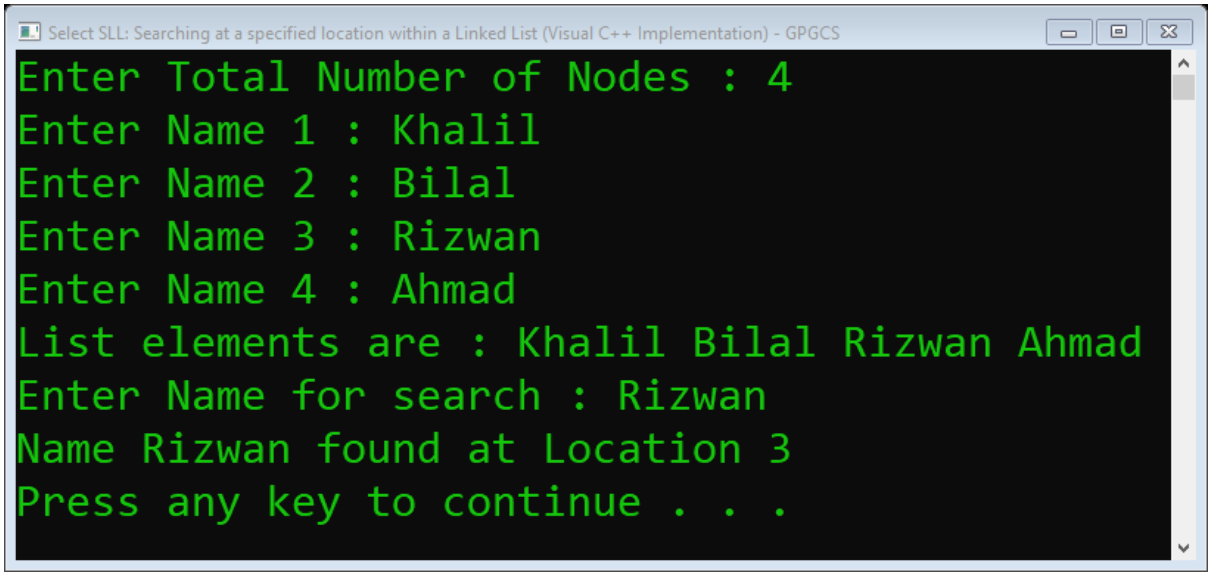
```
                        start = temp;
                    else
                    {
                        current = start;
                        while (current->link != NULL)
                            current = current->link;
                        current->link = temp;
                    }
                }
                // Searching in SLL
                void SearchNode(string _name) {
                    current = start;
                    for (int i = 0; current != NULL ; i++)
                    {
                        if (current->name == _name) {
                            cout << "Name " << current->name << "
                                found at Location " << i+1 << endl;
                            return;
                        }
                        current = current->link;
                    }
                    cout << "Name not found!" << endl;
                }
                // Traversing in SLL
                void DisplayNodes() {
                    current = start;
                    cout << "List elements are : ";
                    while (current != NULL)
                    {
                        cout << current->name << " ";
                        current = current->link;
                    }
                    cout << endl;
                }
        };
}
```

Main.cpp

```
#include <iostream>
#include "SinglyLinkedList.cpp"
using namespace std;
using namespace LinkedList;

int main() {

        SLL obj;

        string name;
        int length;
        cout << "Enter Total Number of Nodes : ";
        cin >> length;
        for (int i = 0; i < length; i++)
        {
            cout << "Enter Name " << i + 1 << " : ";
            cin >> name;
            obj.InsertNodes(name);
        }
        obj.DisplayNodes();
```

```
	cout << "Enter Name for search : ";
	cin >> name;
	obj.SearchNode(name);

	system("pause");
	return 0;
}
```

Output



SLL7 – Deletion at a specified location within a Linked List

(Visual C++ Implementation)

SinglyLinkedList.cpp

```
#include <iostream>
#include <string>
using namespace std;

namespace LinkedList
{
	struct Node
	{
		string name;
		Node *link;
	};

	class SLL
	{
		private:
			Node * start, *temp, *current;
		public:
			SLL() {
				start = NULL;
```

```
                temp = NULL;
                current = NULL;
        }
        // Insertion in SLL
        void InsertNodes(string _name) {
                temp = new Node;
                temp->name = _name;
                temp->link = NULL;
                if (start == NULL)
                        start = temp;
                else
                {
                        current = start;
                        while (current->link != NULL)
                                current = current->link;
                        current->link = temp;
                }
        }
        // Searching in SLL
        void SearchNode(string _name) {
                if (start == NULL) {
                        return;
                }
                current = start;
                int i = 0;
                while(current != NULL)
                {
                        i++;
                        if (current->name == _name) {
                                cout << "Name " << current->name << "
found at Location " << i << endl;
                                return;
                        }
                        current = current->link;
                }
                cout << "Name not found!" << endl;
        }
        // Deletion in SLL
        void DeleteNode(string _name) {
                if (start == NULL) {
                        return;
                }
                current = temp = start;
                if (current->name == _name) {
                        temp = start;
                        start = start->link;
                        delete temp;
                        cout << "Name found and deleted!" << endl;
                        return;
                }
                while(current != NULL)
                {
                        if (current->name == _name) {
                                temp->link = current->link;
                                delete current;
                                cout << "Name found and deleted!" << endl;
                                return;
                        }
                        temp = current;
```

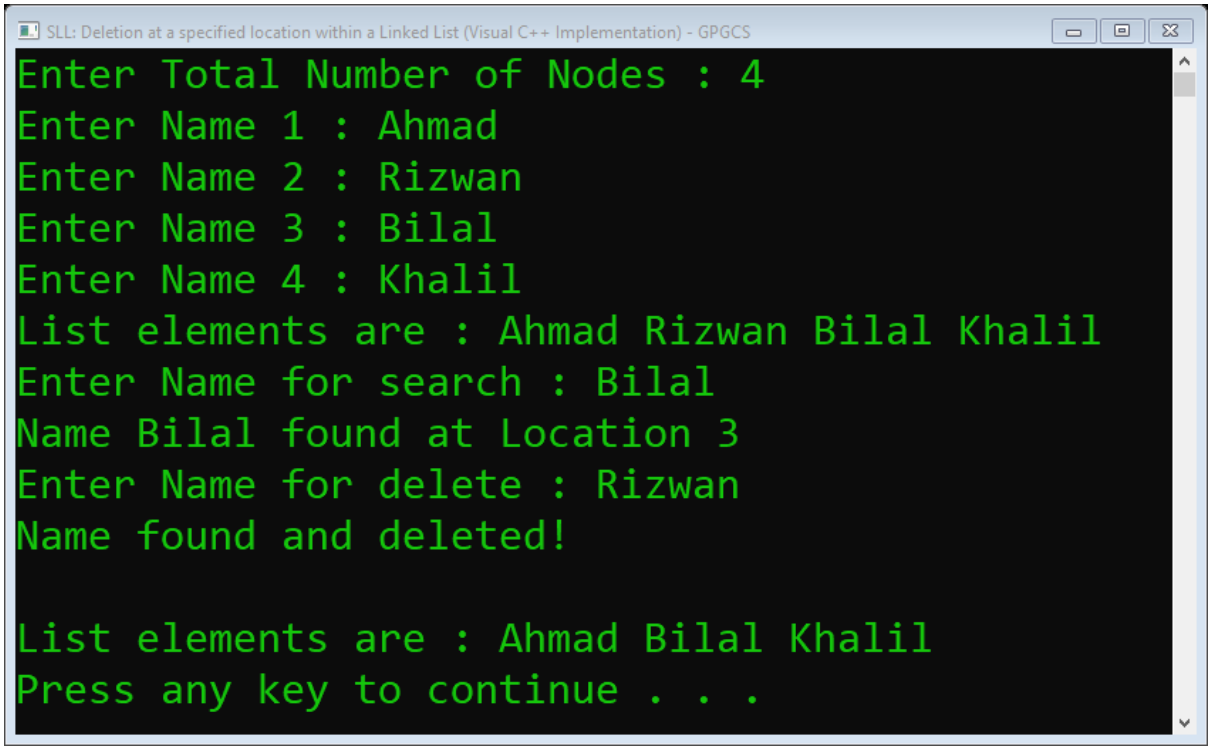
```
                        current = current->link;
                }
                cout << "Name not found!" << endl;
        }
        // Traversing in SLL
        void DisplayNodes() {
                if (start == NULL) {
                        cout << "List is Empty!" << endl;
                        return;
                }
                current = start;
                cout << "List elements are : ";
                while (current != NULL)
                {
                        cout << current->name << " ";
                        current = current->link;
                }
                cout << endl;
        }
    };
}
```

Main.cpp

```
#include <iostream>
#include "SinglyLinkedList.cpp"
using namespace std;
using namespace LinkedList;

int main() {

    SLL obj;

    string name;
    int length;
    cout << "Enter Total Number of Nodes : ";
    cin >> length;

    for (int i = 0; i < length; i++)
    {
        cout << "Enter Name " << i + 1 << " : ";
        cin >> name;
        obj.InsertNodes(name);
    }
    obj.DisplayNodes();

    cout << "Enter Name for search : ";
    cin >> name;
    obj.SearchNode(name);

    cout << "Enter Name for delete : ";
    cin >> name;
    obj.DeleteNode(name);

    cout << "\n";
    obj.DisplayNodes();

    system("\npause");
    return 0;
```

```
}
```

Output



SLL8 – Traversing (Display Nodes) & (Sum of Nodes) (Visual C# Implementation)

LinkedList.cs

```
using System;

namespace SLL_SumOfNodes_CSharp
{
    public class Node
    {
        public int Data { get; set; }
        public Node Next { get; set; }
    }
    public class LinkedList
    {
        private Node start, temp, current;
        public LinkedList() => start = temp = current = null;
        // Insertion in SLL (Add Last)
        public void AddLast(int value)
        {
            // Create Node ...
            temp = new Node
            {
```

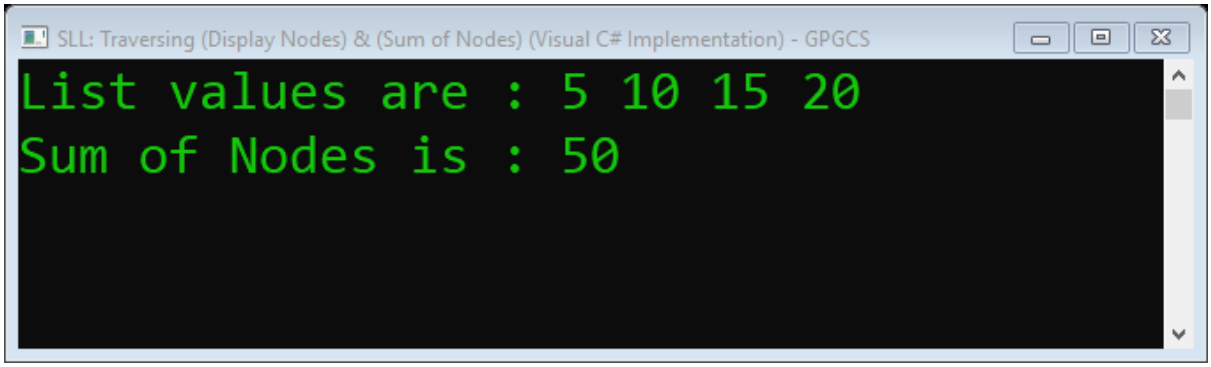
```
                Data = value,
                Next = null
            };

            if (start == null)
            {
                start = temp;
                return;
            }

            // Go to end of linked-list ...
            current = start;
            while (current.Next != null)
                current = current.Next;

            current.Next = temp;
        }

        // Traversing in SLL (Display Nodes)
        public string DisplayList()
        {
            string list = "";
            current = start;
            while(current != null)
            {
                list += current.Data + " ";
                current = current.Next;
            }
            return list;
        }

        // Traversing in SLL (Sum of Nodes)
        public int SumOfNodes()
        {
            int sum = 0;
            current = start;
            while (current != null)
            {
                sum += current.Data;
                current = current.Next;
            }
            return sum;
        }

    }
}
```

Program.cs

```
using System;
using static System.Console;
using static System.Convert;

namespace SLL_SumOfNodes_CSharp
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            LinkedList list = new LinkedList();
            list.AddLast(5);
            list.AddLast(10);
            list.AddLast(15);
            list.AddLast(20);

            WriteLine($"List values are : {list.DisplayList()}");
            WriteLine($"Sum of Nodes is : {list.SumOfNodes()}");

            ReadKey(true);
        }
    }
}
```

Output



SLL9 – Finds Max & Min values from Linked List (Visual C# Implementation)

LinkedList.cs

```
using System;

namespace SLL_Max_Min_Node_CSharp
{
    public class Node
    {
        public int Data { get; set; }
        public Node Next { get; set; }
    }
    public class LinkedList
    {
        private Node start, temp, current;
        public LinkedList() => start = temp = current = null;
        // Insertion in SLL (Add Last)
        public void AddLast(int value)
        {
            // Create Node ...
            temp = new Node
            {
                Data = value,
                Next = null
            };
```

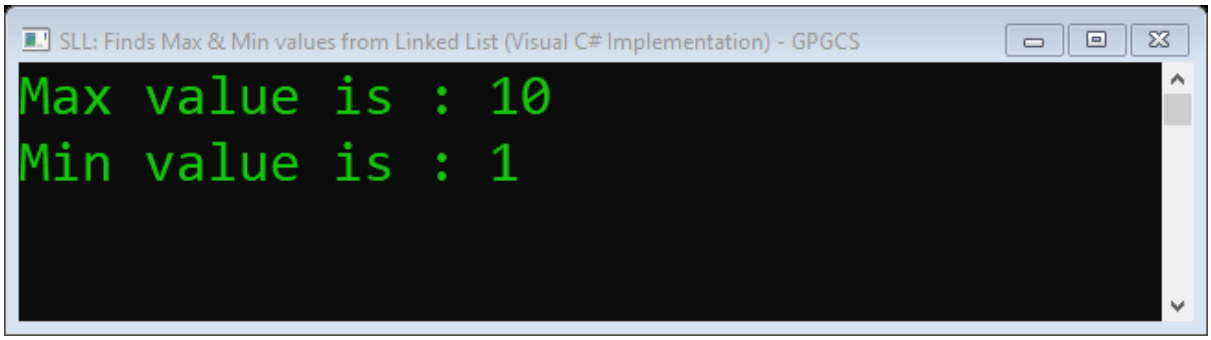
```
            if (start == null)
            {
                start = temp;
                return;
            }

            // Go to end of linked-list ...
            current = start;
            while (current.Next != null)
                current = current.Next;

            current.Next = temp;
        }
        // Find Max value
        public int Max()
        {
            current = start;
            int max = current.Data;
            while(current != null)
            {
                if (current.Data > max)
                    max = current.Data;
                current = current.Next;
            }
            return max;
        }
        // Find Min value
        public int Min()
        {
            current = start;
            int min = current.Data;
            while (current != null)
            {
                if (current.Data < min)
                    min = current.Data;
                current = current.Next;
            }
            return min;
        }
    }
}
```

Program.cs

```
using System;
using static System.Console;

namespace SLL_Max_Min_Node_CSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            LinkedList list = new LinkedList();
            list.AddLast(5);
            list.AddLast(2);
            list.AddLast(10);
            list.AddLast(6);
```

```
            list.AddLast(1);

            WriteLine($"Max value is : {list.Max()}");
            WriteLine($"Min value is : {list.Min()}");

            ReadKey(true);
        }
    }
}
```

Output



Circular Linked-List

Implementation using Visual C++ & Visual C#

CSLL1 – Insertion & Traversing in Circular Single Linked List (Visual C# Implementation)

LinkedList.cs

```
using System;

namespace CSLL1
{
    public class Node
    {
        public int roll_no;
        public string name;
        public string degree;
        public double cgpa;
        public Node link;
    }
    public class LinkedList
    {
        private Node start, temp, current;

        public LinkedList()
        {
            start = temp = current = null;
        }

        // Insertion in CSLL (Insert Node)
        public void InsertNode(int _rollNo, string _name, string _degree, double
_cgpa)
        {
            // Create Node ...
            temp = new Node();
            temp.roll_no = _rollNo;
            temp.name = _name;
            temp.degree = _degree;
            temp.cgpa = _cgpa;

            if(start == null)
            {
                start = temp;
                temp.link = start;  // * Assign the address of starting node in
the pointer field of last node ...
                return;
            }
```

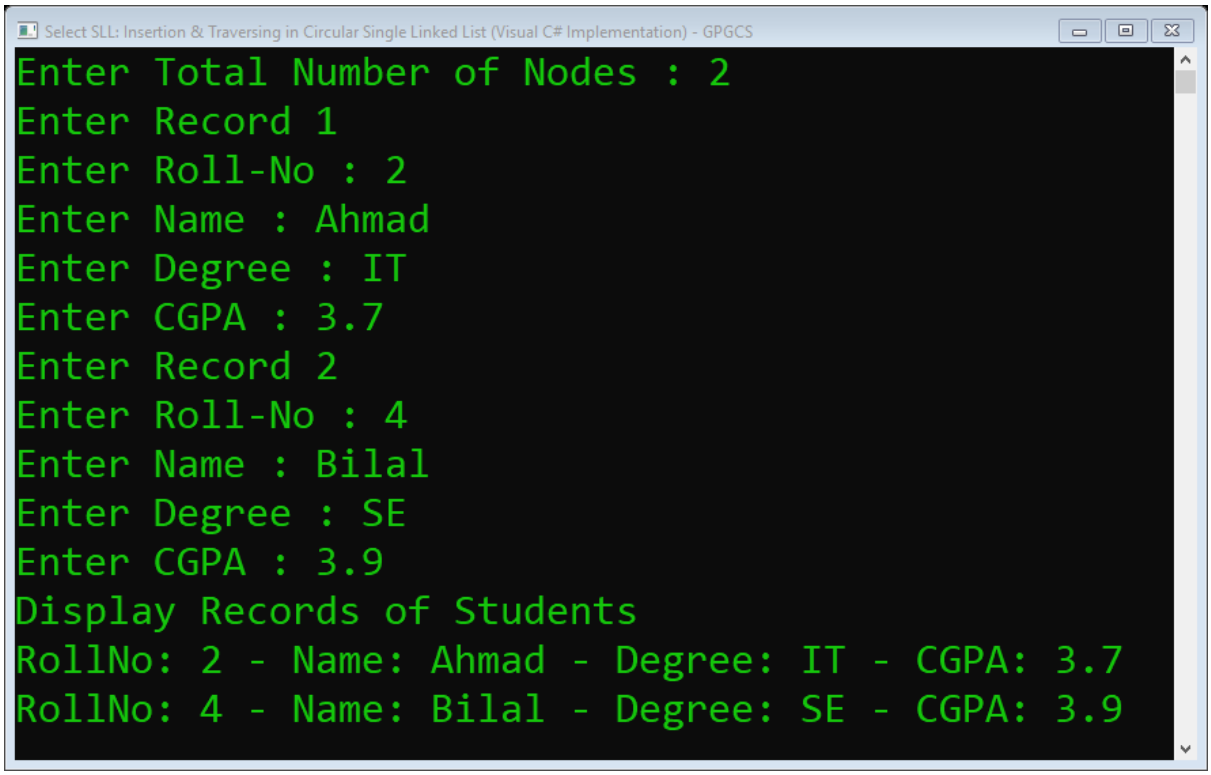
```
            // Go to end of linked-list ...
            current = start;
            while (current.link != start)
                current = current.link;

            current.link = temp;
            temp.link = start;      // * Assign the address of starting node in
the pointer field of last node ...
        }

        // Traversing in CSLL (Display List)
        public string DisplayList()
        {
            string list = "";
            if (start == null)
            {
                list += "List is Empty!";
                return list;
            }
            current = start;
            do
            {
                list += $"RollNo: {current.roll_no} - Name: {current.name} -
Degree: {current.degree} - CGPA: {current.cgpa}\n";
                current = current.link;
            } while (current != start);
            return list;
        }

    }
}
```

Program.cs

```
using System;
using static System.Console;
using static System.Convert;

namespace CSLL1
{
    class Program
    {
        static void Main(string[] args)
        {
            LinkedList list = new LinkedList();

            Write("Enter Total Number of Nodes : ");
            int n = ToInt32(ReadLine());
            for (int i = 0; i < n; i++)
            {
                WriteLine($"Enter Record {i + 1}");

                Write("Enter Roll-No : ");
                int rollNo = ToInt32(ReadLine());
                Write("Enter Name : ");
                string name = ReadLine();
                Write("Enter Degree : ");
                string degree = ReadLine();
                Write("Enter CGPA : ");
```

```
                double cgpa = ToDouble(ReadLine());

                list.InsertNode(rollNo, name, degree, cgpa);
            }

            WriteLine("Display Records of Students");
            WriteLine(list.DisplayList());

            ReadKey(true);
        }
    }
}
```

Output



Double Linked-List

Implementation using Visual C++ & Visual C#

DLL1 – Insertion & Traversing in Double Linked List *

(Visual C# Implementation)

LinkedList.cs

```
using System;

namespace DLL1_CSharp
{
    public class Node
    {
        public int Data { get; set; }
        public Node Next { get; set; }
        public Node Previous { get; set; }
    }
    public class LinkedList
    {
        private Node start, temp, current;

        public LinkedList() => start = temp = current = null;

        // Insertion in DLL (Add Last)
        public void AddLast(int value)
        {
            // Create Node & assign data to it ...
            temp = new Node();
            temp.Previous = null;
            temp.Data = value;
            temp.Next = null;
            if(start == null)
            {
                start = temp;
                return;
            }

            // Go to end of linked-list ...
            current = start;
            while (current.Next != null)
                current = current.Next;

            temp.Previous = current;   // **
            current.Next = temp;
        }
```

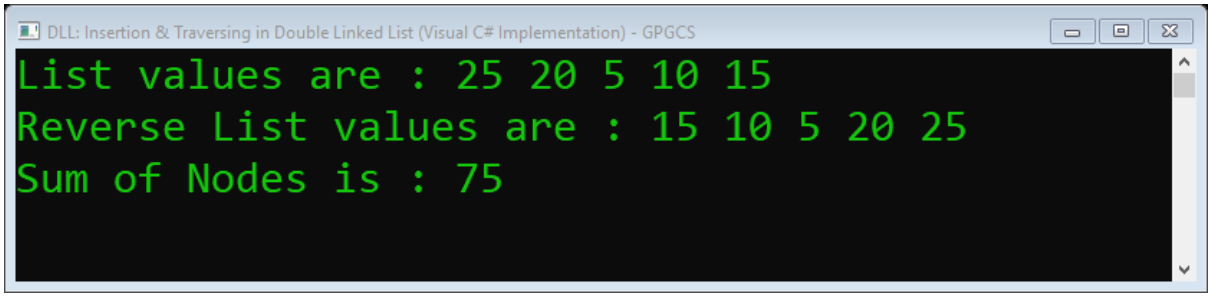
```
// Insertion in DLL (Add First)
public void AddFirst(int value)
{
    // Create Node & assign data to it ...
    temp = new Node();
    temp.Previous = null;
    temp.Data = value;
    temp.Next = null;
    if (start == null)
    {
        start = temp;
        return;
    }

    start.Previous = temp;
    temp.Next = start;
    start = temp;
}
// Traversing in DLL (Display List)
public string DisplayList()
{
    string list = "";
    if (start == null)
    {
        list += "List is Empty!";
        return list;
    }
    current = start;
    // Forward Traversing
    while (current != null)
    {
        list += current.Data + " ";
        current = current.Next;
    }
    return list;
}
// Reverse Traversing in DLL (Reverse Display List)
public string ReverseDisplayList()
{
    string list = "";
    if (start == null)
    {
        list += "List is Empty!";
        return list;
    }
    // Go to end of linked-list ...
    current = start;
    while (current.Next != null)
        current = current.Next;

    // Reverse Traversing
    while (current != null)
    {
        list += current.Data + " ";
        current = current.Previous;
    }

    return list;
}
```

```
        // Traversing in DLL (Sum of Nodes)
        public int SumOfNodes()
        {
            int sum = 0;
            current = start;
            while (current != null)
            {
                sum += current.Data;
                current = current.Next;
            }
            return sum;
        }
    }
}
```

Program.cs

```
using System;
using static System.Console;

namespace DLL1_CSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            LinkedList list = new LinkedList();
            list.AddLast(5);
            list.AddLast(10);
            list.AddLast(15);
            list.AddFirst(20);
            list.AddFirst(25);

            WriteLine($"List values are : {list.DisplayList()}");
            WriteLine($"Reverse List values are : {list.ReverseDisplayList()}");

            WriteLine($"Sum of Nodes is : {list.SumOfNodes()}");


            ReadKey(true);
        }
    }
}
```

Output



DLL – C# LinkedList<T> Class

Program.cs

```
// C# LinkedList<T> Class
// Represents a doubly linked list
// T - Specifies the element type of the linked-list
// Example - Friendship Linked-List

using System;
using System.Collections.Generic;
using System.Linq;
using static System.Console;

namespace LinkedList
{
    class Program
    {
        public static void Display(LinkedList<string> myList)
        {
            foreach (var item in myList)
            {
                Write($"{item} ");
            }
            WriteLine("\n");
        }
        static void Main(string[] args)
        {
            string[] names = {"Ahmad", "Bilal", "Khalil", "Kamal"};

            // Create the linked-list
            LinkedList<string> list = new LinkedList<string>(names);
            WriteLine("The linked list names ...");
            Display(list);

            // Add the name 'Rizwan' to the beginning of the linked-list
            list.AddFirst("Rizwan");
            WriteLine("List 1: Add 'Rizwan' to beginning of the list ...");
            Display(list);

            // Move the first node to be the last node
            LinkedListNode<string> node1 = list.First;
            list.RemoveFirst();
            list.AddLast(node1);
            WriteLine("List 2: Move first node to be last node ...");
            Display(list);

            // Change the last node to 'Anayat'
            list.RemoveLast();
            list.AddLast("Anayat");
            WriteLine("List 3: Change the last node to 'Anayat' ...");
            Display(list);

            // Move the last node to be the first node
            LinkedListNode<string> node2 = list.Last;
            list.RemoveLast();
            list.AddFirst(node2);
```

```
            WriteLine("List 4: Move last node to be first node ...");
            Display(list);

            // LinkedList Methods ..................
            // AddFirst()
            // AddLast()
            // RemoveFirst()
            // RemoveLast()
            // LinkedList Properties ..................
            // First
            // Last

            //...........

            // Add a node 'Rizwan' before the 'Khalil'
            LinkedListNode<string> node3 = list.Find("Khalil");
            list.AddBefore(node3, "Rizwan");
            WriteLine("List 5: Add a node 'Rizwan' before the 'Khalil' ...");
            Display(list);

            // Add nodes 'Faizan' and 'Haris' after the 'Kamal'
            LinkedListNode<string> node4 = list.Find("Kamal");
            list.AddAfter(node4, "Haris");
            list.AddAfter(node4, "Faizan");
            WriteLine("List 6: Add nodes 'Faizan' and 'Haris' after the 'Kamal'
...");
            Display(list);

            // Current node 'Rizwan',
            // and to the previous node & next node in the list.
            LinkedListNode<string> current = list.Find("Rizwan");
            LinkedListNode<string> previous = current.Previous;
            LinkedListNode<string> next = current.Next;
            WriteLine($"Previous Node {previous.Value}, Current Node
{current.Value}, Next Node {next.Value}");

            // Total Nodes in the list
            WriteLine($"\nTotal Friends : {list.Count}\n");

            WriteLine("Friend List <3");
            var friends = from d in list
                          orderby d ascending
                          select d;

            foreach (var friend in friends)
            {
                WriteLine(friend);
            }

            // LinkedList Methods ..................
            // Find()
            // AddBefore()
            // AddAfter()
            // LinkedList Property ..................
            // Count
            // LinkedListNode Properties ..................
            // Next
            // Previous
            // Value
```

```
            // LINQ Query - orderby clause ....................

            ReadKey(true);
        }
    }
}
```

Output

```
DLL – C# LinkedList<T> Class
The linked list names ...
Ahmad Bilal Khalil Kamal

List 1: Add 'Rizwan' to beginning of the list ...
Rizwan Ahmad Bilal Khalil Kamal

List 2: Move first node to be last node ...
Ahmad Bilal Khalil Kamal Rizwan

List 3: Change the last node to 'Anayat' ...
Ahmad Bilal Khalil Kamal Anayat

List 4: Move last node to be first node ...
Anayat Ahmad Bilal Khalil Kamal

List 5: Add a node 'Rizwan' before the 'Khalil' ...
Anayat Ahmad Bilal Rizwan Khalil Kamal

List 6: Add nodes 'Faizan' and 'Haris' after the 'Kamal' ...
Anayat Ahmad Bilal Rizwan Khalil Kamal Faizan Haris

Previous Node Bilal, Current Node Rizwan, Next Node Khalil

Total Friends : 8

Friend List <3
Ahmad
Anayat
Bilal
Faizan
Haris
Kamal
Khalil
Rizwan
```

Self-Task

- Reverse Traversing in SLL & DLL
- Counting Nodes in SLL & DLL
- Display Sum of Odd & Even values in SLL & DLL
- Modify a value from a SLL after searching it

- Insertion at a specified location within a DLL
- Deletion at a specified location within a DLL

https://github.com/MRizwanSE/LinkedList-VisualCplusplus

https://github.com/MRizwanSE/LinkedList-VisualCSharp

Best of Luck 😊