Department of IT and Computer Science
Pak-Austria Fachhochschule: Institute of Applied Sciences
and Technology, Haripur, Pakistan

COMP-201L Data Structures and Algorithms Lab

Lab Report: 06

Class: Computer Science

Name: Yaseen Ejaz Ahmed

Registration No.: B20F0283CS014

Semester: Third

Submitted to: Engr. Rafi Ullah

_____

Instructor Signature

Lab No. 6

Single link list, Double link list, Link list operations

Objectives:

- To understand Single and double link lists.
- To Implement link lists in C++.

Tools/Software Required:

C++ Compiler

Introduction:

A linked list is just a chain of nodes, with each subsequent node being a child of the previous one. Many programs rely on linked lists for their storage because these don't have any evident restrictions. For example, the array list we did earlier could not grow or shrink, but node-based ones can! This means there is no limit (other than the amount of memory) on the number of elements they can store.

Lab Tasks:

NOTE: Please find the CPP files attached.

Lab Task 01: Write and test a method public int size ( ) to count the number of nodes in the linked list.

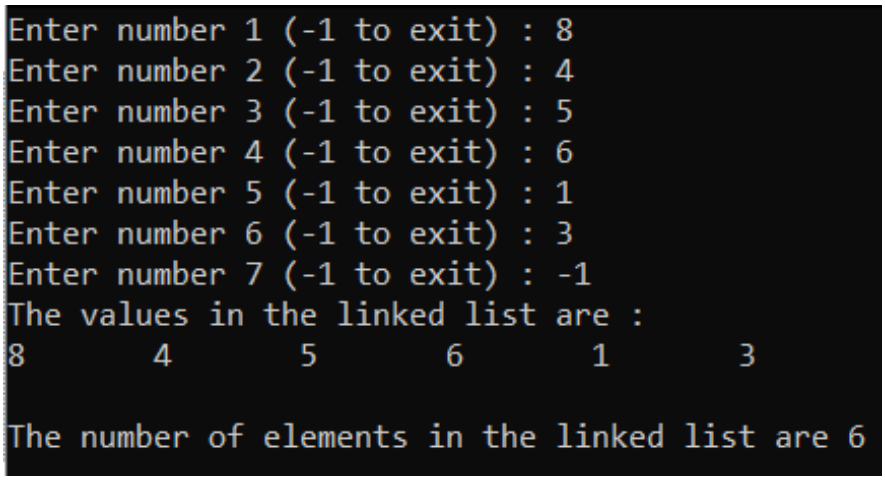
Code:

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

class node
{
        private:
        int data;
        node* next;
        node *head=NULL;
        node *ptr;

        public:
        void InsertNode(int value)
        {
                node* temp=new node();
                temp->data=value;

                temp->next=NULL;

                if(head==NULL)
                {
                        head=temp;
                        ptr=head;
                }

                else
                {
                        ptr->next=temp;
                        ptr=ptr->next;
                }
        }
```

```
        void show()
        {
                ptr=head;
                cout<<"\n\nThe values in the linked list are :\n";
                while(ptr->next!=NULL)
                {
                        cout<<ptr->data<<"\t";
                        ptr=ptr->next;
                }
                cout<<ptr->data;
        }

        int size()
        {
                int size=0;
                ptr=head;
                cout<<endl;
                while(ptr->next!=NULL)
                {
                        ptr=ptr->next;
                        size++;
                }
                size++;
                return size;
        }

};

int main()
{
        int value,num,i=1;
        node n;
        do
        {
                cout<<"Enter number "<<i<<" (-1 to exit) : ";
                cin>>value;

                if(value!=-1)
                n.InsertNode(value);
                i++;
        }
```

```
        while(value!=-1);
        n.show();
        cout<<"\nThe number of elements in the linked list are "<<n.size();
}
```

Output:



Lab Task 02: Write and test a method public Boolean search (int n) to find out whether the given data exists or not in the linked list.

Code:

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

class node
{
        private:
        int data;
        node* next;
        node *head=NULL;
        node *ptr;

        public:
        void InsertNode(int value)
        {

                node* temp=new node();
```

```
        temp->data=value;
        temp->next=NULL;

        if(head==NULL)
        {
                head=temp;
                ptr=head;
        }

        else
        {
                ptr->next=temp;
                ptr=ptr->next;
        }
}

void show()
{
        ptr=head;
        cout<<"\n\nThe values in the linked list are :\n";
        while(ptr->next!=NULL)
        {
                cout<<ptr->data<<"\t";
                ptr=ptr->next;
        }

        cout<<ptr->data;
}

bool find(int find)
{
        int found;
        ptr=head;
        while(ptr->next!=NULL)
        {
                if(ptr->data == find)
                return true;

                ptr=ptr->next;
        }
```

```
            if(ptr->data == find)
            return true;
            return false;
        }

};

int main()
{
        int value,search,i=1;
        node n;
        do
        {
            cout<<"Enter number "<<i<<" (-1 to exit) : ";
            cin>>value;

            if(value!=-1)
            n.InsertNode(value);
            i++;
        }
        while(value!=-1);

        n.show();
        cout<<"\n\nEnter the number to find : ";
        cin>>search;

        bool flag=n.find(search);

        if(flag)
        cout<<"\nIt is present in the linked list";

        else
        cout<<"\nIt is NOT present in the linked list";
}
```

Output:

```
Enter number 1 (-1 to exit) : 1
Enter number 2 (-1 to exit) : 2
Enter number 3 (-1 to exit) : 3
Enter number 4 (-1 to exit) : 4
Enter number 5 (-1 to exit) : 5
Enter number 6 (-1 to exit) : -1


The values in the linked list are :
1       2       3       4       5

Enter the number to find : 3

It is present in the linked list
```

Lab Task 03: Swap nodes in a linked list without swapping data.

Code:

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

class node
{
        private:
        int data;
        node* next;
        node *head=NULL;
        node *ptr;

        public:
        void InsertNode(int value)
        {
                node* temp=new node();
                temp->data=value;
                temp->next=NULL;
```

```
        if(head==NULL)
        {
                head=temp;
                ptr=head;
        }

        else
        {
                ptr->next=temp;
                ptr=ptr->next;
        }
}

void show()
{
        ptr=head;
        cout<<"\n\nThe values in the linked list are :\n";
        while(ptr->next!=NULL)
        {
                cout<<ptr->data<<"\t";
                ptr=ptr->next;
        }

        cout<<ptr->data;
}

bool find(int find)
{
        int found;
        ptr=head;
        while(ptr->next!=NULL)
        {
                if(ptr->data == find)
                return true;

                ptr=ptr->next;
        }

        if(ptr->data == find)
        return true;

        return false;
```

```
        }


        void swaping(int num1,int num2)
        {
                ptr=head;
                while(ptr->data!=num1)
                {
                        ptr=ptr->next;
                }

                node*temp=head;
                while(temp->data!=num2)
                {
                        temp=temp->next;
                }
                temp->data=num1;
                ptr->data=num2;
        }
};

int main()
{
        int value,no,search,i=1;
        int num1,num2;
        node n;

        do
        {
                cout<<"Enter number "<<i<<" (-1 to exit) : ";
                cin>>value;

                if(value!=-1)
                n.InsertNode(value);
                i++;
        }
        while(value!=-1);

        n.show();

        cout<<"\n\nEnter the first number to find : ";
        cin>>num1;
```

```
        bool flag1=n.find(num1);

        if(flag1)
        cout<<"\nFirst number present in the linked list";
        else
        cout<<"\nFirst number is NOT present in the linked list";


        cout<<"\n\nEnter the second number to find : ";
        cin>>num2;

        bool flag2=n.find(num2);

        if(flag2)
        cout<<"\nSecond number present in the linked list";
        else
        cout<<"\nSecond number is NOT present in the linked list";


        if(flag1 && flag2)
        {
                n.swaping(num1,num2);
        }
        n.show();
}
```

Output:

```
Enter number 1 (-1 to exit) : 1
Enter number 2 (-1 to exit) : 2
Enter number 3 (-1 to exit) : 3
Enter number 4 (-1 to exit) : 4
Enter number 5 (-1 to exit) : 5
Enter number 6 (-1 to exit) : -1


The values in the linked list are :
1       2       3       4       5

Enter the first number to find : 2

First number present in the linked list

Enter the second number to find : 4

Second number present in the linked list

The values in the linked list are :
1       4       3       2       5
--------------------------------
```

Home Tasks:

Q1: Write a function that counts the number of times a given int occurs in a Linked List.

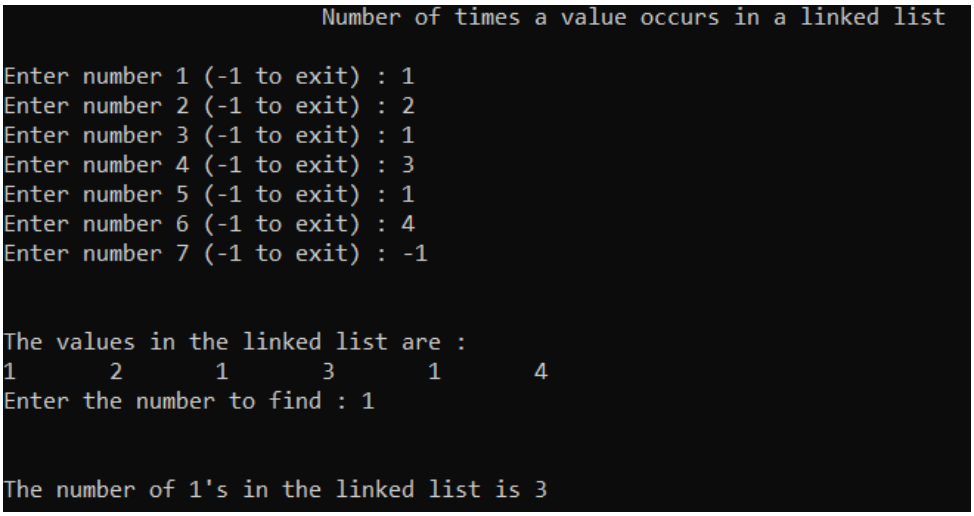
Code:

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

class node
{
        private:
        int data;
        node* next;
        node *head=NULL;
        node *ptr;

        public:
        void InsertNode(int value)
        {

                node* temp=new node();
                temp->data=value;

                temp->next=NULL;

                if(head==NULL)
                {
                        head=temp;
                        ptr=head;
                }

                else
                {
                        ptr->next=temp;
                        ptr=ptr->next;
                }
        }
```

```
        void show()
        {
                ptr=head;
                cout<<"\n\nThe values in the linked list are :\n";
                while(ptr->next!=NULL)
                {
                        cout<<ptr->data<<"\t";
                        ptr=ptr->next;
                }

                cout<<ptr->data;
        }

        int count(int n)
        {
                int counter=0;
                ptr=head;
                while(ptr->next!=NULL)
                {
                        if(ptr->data == n)
                        counter++;

                        ptr=ptr->next;
                }

                if(ptr->data == n)
                counter++;
                return counter;
        }

};

int main()
{
        int value,search,i=1;
        node n;
        cout<<"\t\t\tNumber of times a value occurs in a linked list\n\n";
        do
        {
                cout<<"Enter number "<<i<<" (-1 to exit) : ";
                cin>>value;
```

```
            if(value!=-1)
            n.InsertNode(value);
            i++;
        }
        while(value!=-1);

        n.show();
        cout<<"\nEnter the number to find : ";
        cin>>search;

        cout<<"\n\nThe number of "<<search<<"'s in the linked list is "<<n.count(search);
}
```

Output:



Q2: Write a function Linked list traversal using recursion in c++.

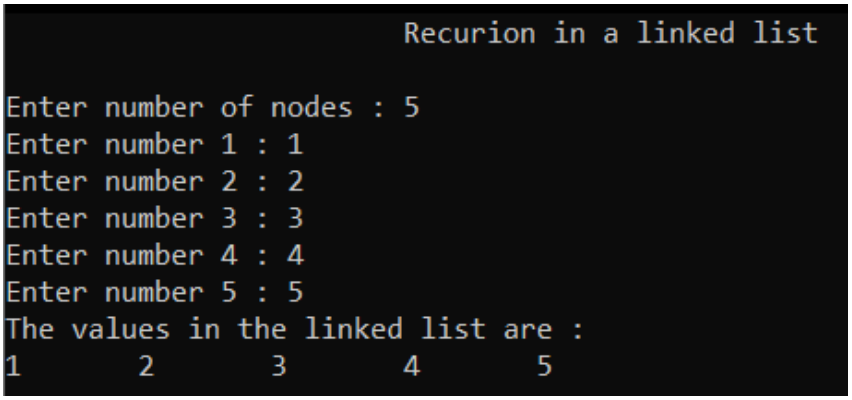
Code:

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

class node
{
        private:
        int data;
        node* next;
        node *head=NULL;
```

```
node *ptr;

public:
void InsertNode(int value[],int index,int size)
{
        node* temp=new node();
        temp->data=value[index];

        temp->next=NULL;

        if(head==NULL)
        {
                head=temp;
                ptr=head;
        }

        else
        {
                ptr->next=temp;
                ptr=ptr->next;
        }
        index++;

        if(index<=size)
        InsertNode(value,index,size);       //recursion
}

void initialize()
{
        ptr=head->next;
        cout<<"The values in the linked list are :\n";
}

void show()
{
        if(ptr->next!=NULL)
        {
                cout<<ptr->data<<"\t";
                ptr=ptr->next;
                show();          //recursion
        }
        else
```

```
            cout<<ptr->data<<"\t";

        }

};

int main()
{
        int value,num;
        node n;
        cout<<"\t\t\tRecursion in a linked list\n\n";

        cout<<"Enter number of nodes : ";
        cin>>num;
        int a[num];
        for(int i=1;i<=num;i++)
        {
            cout<<"Enter number "<<i<<" : ";
            cin>>a[i];
        }
        int start=0;
        n.InsertNode(a,start,num);
        n.initialize();
        n.show();
        delete[] a;
}
```

Output:



Q3: There is a garage where the access road can accommodate any number of trucks at one time. The garage is build such a way that only the last truck entered can be moved out. Each of the trucks is identified by a positive integer (a truck_id). Write a program to handle truck moves, allowing for the following commands:

a) On_road (truck_id);
b) Enter_garage (truck_ id);
c) Exit_garage (truck_id);
d) Show_trucks (garage or road);

If an attempt is made to get out a truck which is not the closest to the garage entry, the error
message Truck x not near garage door.

Code:

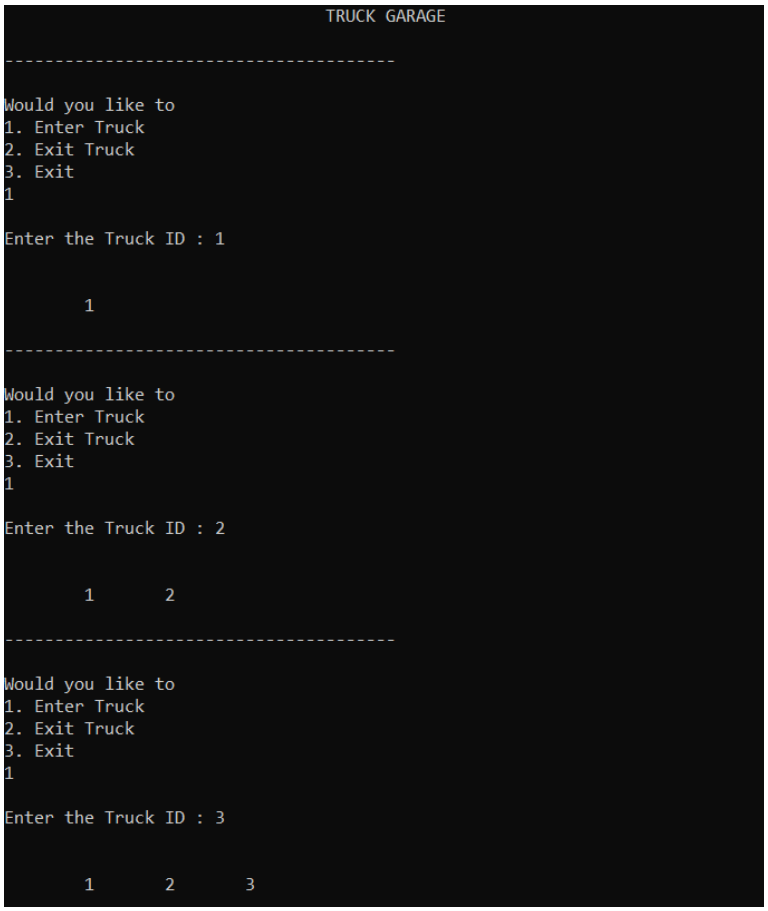
```
#include <iostream>
using namespace std;

class node
{
        private:
        int data;
        node* next;
        node *head=NULL;
        node *ptr;

        public:
        void InsertTruck(int truck_ID)
        {
                node* temp=new node();
                temp->data=truck_ID;
                temp->next=NULL;

                if(head==NULL)
                {
                        head=temp;
                        ptr=head;
                }
```

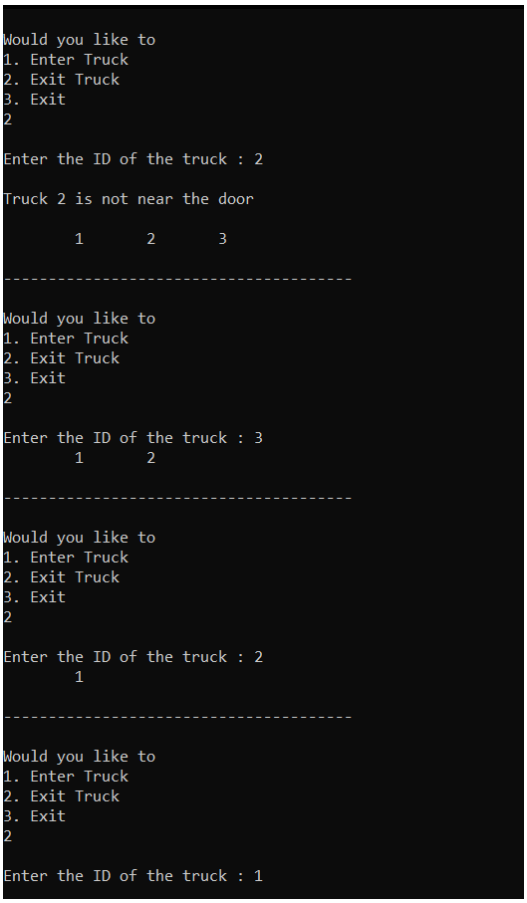
```
        else
        {
                ptr->next=temp;
                ptr=ptr->next;
        }
}

void show()
{
        ptr=head;

        while(ptr->next!=NULL)
        {
                cout<<"\t"<<ptr->data;
                ptr=ptr->next;
        }

        cout<<"\t"<<ptr->data;
}

bool search(int ID)
{
        ptr=head;

        while(ptr->next!=NULL)
        {

                if(ptr->data==ID)
                return true;
                ptr=ptr->next;
        }

        if(ptr->data==ID)
        return true;

        return false;
}

int RemoveTruck()
{
        int ID;
        cout<<"\nEnter the ID of the truck : ";
```

```
                cin>>ID;

                if(search(ID))
                {
                        ptr=head;

                   while(ptr->next->next!=NULL)
                   {
                        ptr=ptr->next;
                   }

                   node* temp=ptr->next;

                   if(temp->data==ID)
                {
                        delete temp;
                   ptr->next=NULL;
                }

                   else cout<<"\nTruck "<<ID<<" is not near the door\n\n";
                }

                else return 0;
        }

};

int main()
{
        node n;
        int opt,truck_ID;
        cout<<"\t\t\t\tTRUCK GARAGE";
        do
        {
                cout<<"\n\n---------------------------------------\n\nWould you like to\n1. Enter
Truck\n2. Exit Truck\n3. Exit\n";
                cin>>opt;

                if(opt==1)
                {
                        cout<<"\nEnter the Truck ID : ";
                        cin>>truck_ID;
```

```
                n.InsertTruck(truck_ID);
                cout<<endl<<endl;
                n.show();
        }

        else if(opt==2)
        {
                n.RemoveTruck();
                n.show();
        }
    }
    while(opt!=3);
}
```

Output:





Q4: Simulate a game, using two circular doubly linked list. The game involves a number of children, and you are supposed to read their names from a file. The children whose names are on lines numbered by prime numbers should be placed on the first list, and the others on the second list. Starting with the child whose name is on the line in the middle (or ⌊numberOfChildren/2⌋) of the second list, children on that list are counted clockwise. Every mth child, where m is the number of elements in the first list is eliminated from that list. Counting goes on with the next child. Repeat this counting m times or till the second list gets empty. Your program should output the initial lists and the final second list.

Deleting From Second List and Adding to First:

Code:

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

class node
{
        private:
        string data;
```

```
node* next;
node* pre;
node *head=NULL;
node *ptr;

public:
void InsertNode(string value,int flag)
{
        node* temp=new node();
        temp->data=value;
        temp->pre=NULL;
        temp->next=NULL;

        if(head==NULL)
        {
                head=temp;
                ptr=head;
        }

        else if(flag==0)
        {
                ptr->next=temp;
                temp->pre=ptr;
                ptr=temp;
        }

        else if(flag==1)
        {
                ptr=head->pre;
                ptr->next=temp;
                temp->pre=ptr;

                temp->next=head;
                head->pre=temp;
        }
}

void Circular()
{
        ptr->next=head;
        head->pre=ptr;
}
```

```
void FindNode(int num)
{
	num=num/2;
	ptr=head;
	int pivot=1;
	while(pivot<num)
	{
		pivot++;
		ptr=ptr->next;
	}

	cout<<"\nMID VALUE : "<<num;
	cout<<"\nSTARTING VALUE : "<<ptr->data;

}

string RemoveNode(int num)
{
	string s;
	cout<<"\nVALUE TO REMOVE VALUE : ";
	for (int i=1 ; i<=num ; i++)
	{
		cout<<ptr->data<<" -> ";
		ptr=ptr->next;
	}
	cout<<ptr->data;
	cout<<"\nCLOCKWISE : "<<num<<endl<<endl;

	if(ptr!=head)
	{
		node *temp1 = ptr->pre;
	node *temp2= ptr->next;

	temp1->next=temp2;
	temp2->pre=temp1;
	s=ptr->data;
	}

	else if(ptr==head)
	{
		node*temp=head;
		head=head->next;
```

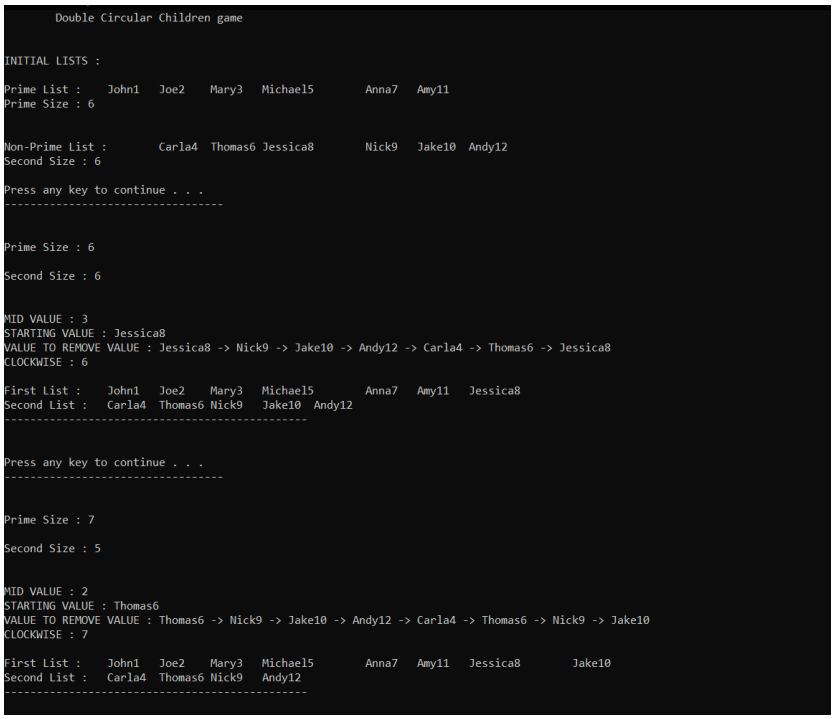
```
                ptr=ptr->pre;

                ptr->next=head;
                head->pre=ptr;
                s=temp->data;
        }
        return s;


}

bool isEmpty()
{
        if(head->next==head)
        {
                head=NULL;
                return true;
        }

        else return false;
}

void show()
{
        ptr=head;
        do
        {
                cout<<"\t"<<ptr->data;
                ptr=ptr->next;
        }
        while(ptr!=head);
}

int size()
{
        int size=0;
        ptr=head;
        cout<<endl;
        do
        {
                ptr=ptr->next;
                size++;
```

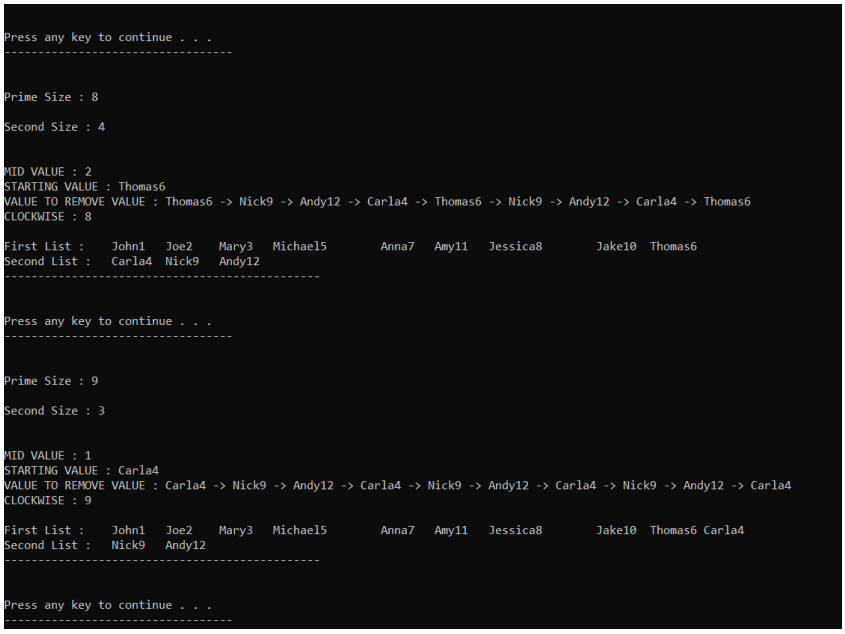
```
            }

            while(ptr!=head);
            return size;
        }

};

int main()
{
        node FirstList,SecondList;
        string s;
        int num=0;
        bool flag;
        fstream file("name.txt");

        cout<<"\tDouble Circular Children game\n";
        while(!file.eof())
        {
            flag=false;
            getline(file,s);
            num++;

            for(int i=2;i<num;i++)
            {
                if(num%i==0)
                {
                    flag=true;
                    break;
                }
            }

            if(!flag)
            {
               FirstList.InsertNode(s,0);
            }

            else
            {
                SecondList.InsertNode(s,0);
            }
        }
```

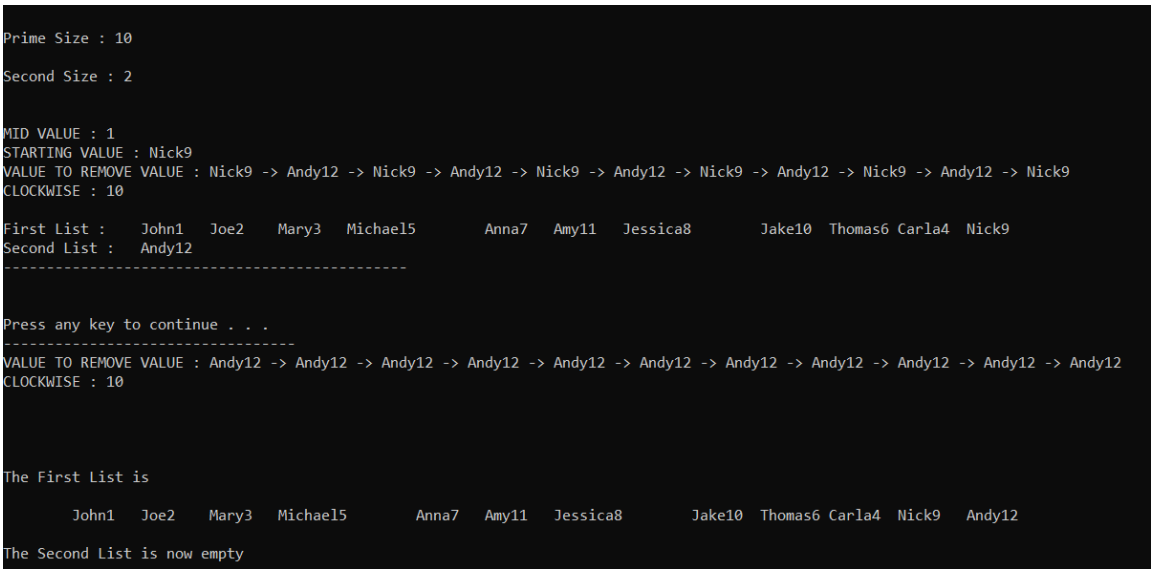
```
FirstList.Circular();
SecondList.Circular();
cout<<"\n\nINITIAL  LISTS :\n\n";
cout<<"Prime List : ";
FirstList.show();
int size1=FirstList.size();
cout<<"Prime Size : "<<size1;
cout<<endl<<endl;
cout<<"\nNon-Prime List : ";
SecondList.show();
int size2=SecondList.size();
cout<<"Second Size : "<<size2;

cout<<endl<<endl;
system("pause");
cout<<"---------------------------------";


while(!SecondList.isEmpty())
{
	size1=FirstList.size();
	size2=SecondList.size();
	cout<<"\nPrime Size : "<<size1<<endl;
	cout<<"\nSecond Size : "<<size2<<endl;
	cout<<endl;
	SecondList.FindNode(size2);
	string s=SecondList.RemoveNode(size1);

	FirstList.InsertNode(s,1);
	cout<<"First List : ";
	FirstList.show();
	cout<<"\nSecond List : ";
	SecondList.show();
	cout<<"\n-------------------------------------------";
	size2=SecondList.size();
	cout<<endl<<endl;
	system("pause");
	cout<<"---------------------------------";
}

s=SecondList.RemoveNode(size1);
FirstList.InsertNode(s,1);
```

```
    cout<<"\n\n\nThe First List is\n\n";
    FirstList.show();
    cout<<"\n\nThe Second List is now empty";

}
```

Output:







Only Deleting From Second List:

Code:

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

class node
{
        private:
        string data;
        node* next;
        node* pre;
        node *head=NULL;
        node *ptr;

        public:
        void InsertNode(string value)
        {
                node* temp=new node();
                temp->data=value;
                temp->pre=NULL;
                temp->next=NULL;

                if(head==NULL)
                {
                        head=temp;
```

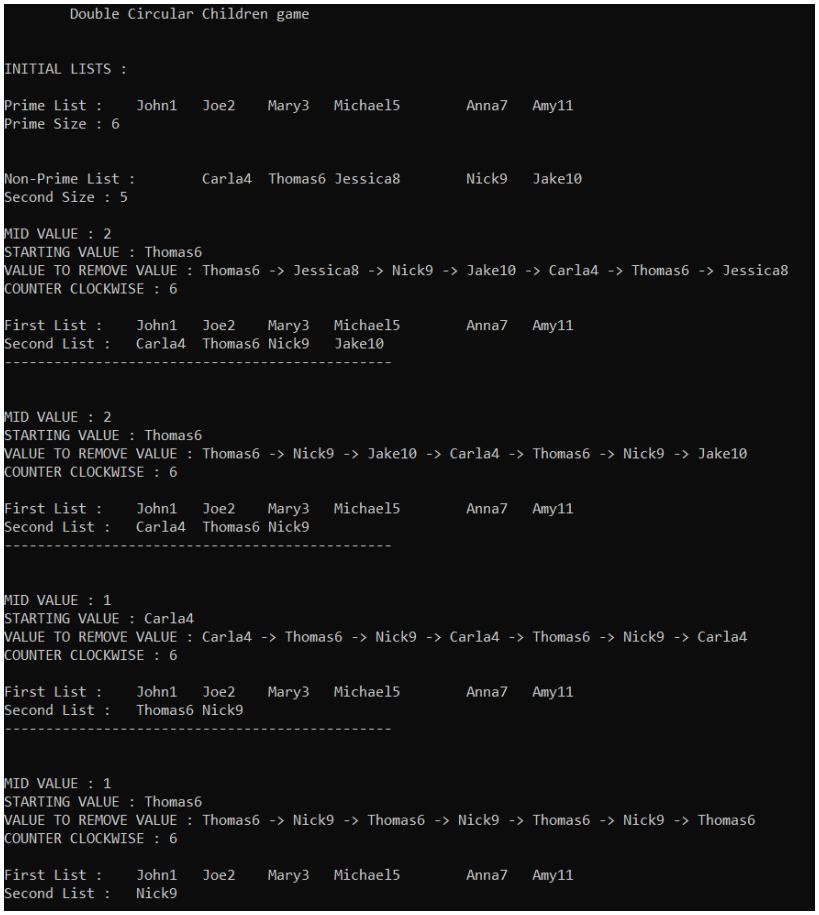
```
            ptr=head;
        }

        else
        {
            ptr->next=temp;
            temp->pre=ptr;
            ptr=temp;
        }
}

void Circular()
{
        ptr->next=head;
        head->pre=ptr;
}

void FindNode(int num)
{
        num=num/2;
        ptr=head;
        int pivot=1;
        while(pivot<num)
        {
            pivot++;
            ptr=ptr->next;
        }

        cout<<"\nMID VALUE : "<<num;
        cout<<"\nSTARTING VALUE : "<<ptr->data;

}

void RemoveNode(int num)
{
        cout<<"\nVALUE TO REMOVE VALUE : ";
        for (int i=1 ; i<=num ; i++)
        {
            cout<<ptr->data<<" -> ";
            ptr=ptr->next;
        }
        cout<<ptr->data;
```

```
        cout<<"\nCLOCKWISE : "<<num<<endl<<endl;

        if(ptr!=head)
        {
                node *temp1 = ptr->pre;
        node *temp2= ptr->next;

        temp1->next=temp2;
        temp2->pre=temp1;
        }

        else if(ptr==head)
        {
                node*temp=head;
                head=head->next;
                ptr=ptr->pre;

                ptr->next=head;
                head->pre=ptr;
        }
}

bool isEmpty()
{
        if(head->next==head)
        {
                head=NULL;
                return true;
        }


        else return false;
}

void show()
{
        ptr=head;
        do
        {
                cout<<"\t"<<ptr->data;
                ptr=ptr->next;
        }
```

```
            while(ptr!=head);
        }
        int size()
        {
            int size=0;
            ptr=head;

            cout<<endl;


            do
            {
                ptr=ptr->next;
                size++;
            }
            while(ptr!=head);
            return size;
        }

};


int main()
{
        node FirstList,SecondList;
        string s;
        int num=0;
        bool flag;
        fstream file("name.txt");
        cout<<"\tDouble Circular Children game\n";
        while(!file.eof())
        {
            flag=false;
            getline(file,s);
            num++;

            for(int i=2;i<num;i++)
            {
                if(num%i==0)
                {
                    flag=true;
                    break;
```

```
                }
            }

            if(!flag)
            {
               FirstList.InsertNode(s);
            }

            else
            {
                SecondList.InsertNode(s);
            }
        }

        FirstList.Circular();
        SecondList.Circular();
        cout<<"\n\nINITIAL  LISTS :\n\n";
        cout<<"Prime List : ";
        FirstList.show();
        int size1=FirstList.size();
        cout<<"Prime Size : "<<size1;
        cout<<endl<<endl;
        cout<<"\nNon-Prime List : ";
        SecondList.show();
        int size2=SecondList.size();
        cout<<"Second Size : "<<size2;


        while(!SecondList.isEmpty())
        {
            cout<<endl;
            SecondList.FindNode(size2);
            SecondList.RemoveNode(size1);

            cout<<"First List : ";
            FirstList.show();
            cout<<"\nSecond List : ";
            SecondList.show();
            cout<<"\n-----------------------------------------";
            size2=SecondList.size();
        }
}
```

Output:



Results & Observations:

Link Lists can be used for undefined number of memory locations instead of fixed value such as arrays. We can traverse through link list can be used through pointers by knowing the address of the next data. We can add new values as well as delete values.