```
/*
Name: Shubham Sarang
Roll no: 1345
Unit 5: Linked list
Program: Singly Linked List
*/

#include<iostream>
#include<conio.h>
using namespace std;

//Node template
class Node
{
	public:
		int data;
		Node *next;
};

//List tempate
class SList
{
	Node *head;

	public:
		SList()
		{
			head=NULL;
		}
		void Insert(int x);
		void Display();
		void Length();
		void Search(int x);
		void Remove(int x);
};
// Functions
void SList :: Insert(int x)
{
	//create node t
	Node *t= new Node();
	t->data=x;
	t->next = NULL;

	//first node in the list
	if(head == NULL)
```

```
        {
                head = t;
                return;
        }

        //traverse till last node
        Node *tmp = head;

        while(tmp->next != NULL)
        {
                tmp= tmp->next;
        }
        //attach t to the last node tmp
        tmp->next = t;
}
void SList :: Display()
{
        Node *tmp = head;
        while(tmp!=NULL)
        {
                cout<<tmp->data<<"->";
                tmp = tmp->next;
        }
        cout<<"End of list";
}

void SList :: Length()
{
        Node *tmp = head;
        int count=0;
        while(tmp!=NULL)
        {
                count++;
                tmp = tmp->next;
        }
        cout<<"Number of elements in list is "<<count;
}

void SList :: Search(int x)
{
        Node *tmp=head;
        int flag=0;

        while(tmp!=NULL)
        {
```

```
            if(tmp->data==x)
            {
                flag=1;
                break;
            }
            tmp=tmp->next;
        }

        if(flag==1)
        {
            cout<<"Element "<<x<<" found";
        }
        else
        {
            cout<<"Element "<<x<<" not found";
        }
}

void SList :: Remove(int x)
{
        //Step 1:List in empty return control
        if(head==NULL)
        {
            cout<<"LIST IS EMPTY";
            return;
        }
        //Step 2: Search element
        Node *tmp = head;
        Node *prev = NULL;
        int flag = 0;

        while(tmp!=NULL)
        {
            if(tmp->data == x)
            {
                flag = 1;
                break;
            }
            prev = tmp;
            tmp = tmp->next;
        }

        //step 3: unsuccessfull search return control
        if(flag==0)
        {
```

```
            cout<<"Element "<<x<<" not found ";
            return;
        }

        //Step 4: Successful search, a single node deletion
        if(tmp == head && tmp->next == NULL)
        {
            head = NULL;
        }
        else if(tmp==head) //step 4 b: head node deletion
        {
            head = tmp->next;
        }
        else if(tmp->next == NULL) //step4 c: tail node deletion
        {
            prev->next = NULL;
        }
        else //Step 4 d: any node in the middle
        {
            prev->next = tmp->next;
        }
    //Step 5: delete the node containing x
    delete tmp;
    cout<<"Element deleted ";
}
//Menu

int main()
{
        int ch, num, y;
        SList s;
        while(1)
        {
            system("cls");
            cout<<"**Singly linked list**"<<endl;
            cout<<"1. Insert in SLL\n";
            cout<<"2. Display List\n";
            cout<<"3. Length of SLL\n";
            cout<<"4. Search for the node in SLL\n";
            cout<<"5. Remove a node\n";
            cout<<"6. Exit\n\n";

            cout<<"Enter your choice: ";
            cin>>ch;
```

```
		switch(ch)
		{
			case 1:
					cout<<"Enter the data: ";
					cin>>num;
					s.Insert(num);
					getch();
					break;
			case 2:
					s.Display();
					getch();
					break;
			case 3:
					s.Length();
					getch();
					break;
			case 4:
					cout<<"Enter the element to be searched: ";
					cin>>y;
					s.Search(y);
					getch();
					break;
			case 5:
					cout<<"Enter the element to be removed: ";
					cin>>y;
					s.Remove(y);
					s.Display();
					getch();
					break;
			case 6:
					exit(1);
			default:
					cout<<"Incorrect option";
					getch();
		}//end of switch


	}//end while

}//end main
```

output:

```
C:\Geralt\DSL\Searching Unit 2\SLL\SLL.exe
**Singly linked list**
1. Insert in SLL
2. Display List
3. Length of SLL
4. Search for the node in SLL
5. Delete a node
6. Exit

Enter your choice:
```

Enter Elements:

```
C:\Geralt\DSL\Searching Unit 2\SLL\SLL.exe
**Singly linked list**
1. Insert in SLL
2. Display List
3. Length of SLL
4. Search for the node in SLL
5. Delete a node
6. Exit

Enter your choice: 1
Enter the data: 23
```

Display elements:

```
C:\Geralt\DSL\Searching Unit 2\SLL\SLL.exe
**Singly linked list**
1. Insert in SLL
2. Display List
3. Length of SLL
4. Search for the node in SLL
5. Delete a node
6. Exit

Enter your choice: 2
23->25->28->30->End of list
```

Length of list:

```
Enter your choice: 3
Number of elements in list is 4
```

Search element:

```
Enter your choice: 4
Enter the element to be searched: 23
Element 23 found
```

```
Enter your choice: 4
Enter the element to be searched: 10
Element 10 not found
```

Head Node Deletion:

```
Enter your choice: 2
23->24->25->26->27->28->End of list
```

```
Enter your choice: 5
Enter the element to be removed: 23
Element deleted 24->25->26->27->28->30->End of list
```

Tail Node Deletion:

```
Enter your choice: 5
Enter the element to be removed: 30
Element deleted 24->25->26->27->28->End of list
```

Any Node Deletion:

```
Enter your choice: 5
Enter the element to be removed: 26
Element deleted 24->25->27->28->End of list
```

Program:

```
/*
Name: Shubham Sarang
Roll no: 1345
Unit 5: Linked list
Program: Circular Linked List
*/

#include<iostream>
#include<conio.h>
using namespace std;

//Node template
class CNode
{
	public:
		int data;
		CNode *next; //declare data to store value and create node

};

//list template
class CList
{
	CNode *first; //first node
```

```
        CNode *last;       //last node

        public:
                CList()
                {
                        first = NULL;
                        last = NULL;
                }
                void Insert(int x); //Methods
                void Display();
                void Length();
                void Search(int x);
                void Remove(int x);
};

//functions

void CList :: Insert(int x)
{
        CNode *t = new CNode();  //temporary node to store value
        t->data = x;
        t->next = NULL;

        if(first == NULL && last == NULL)  //head node
        {
                first = t;
                last = t;
                last->next = first;
        }
        else
        {
                last->next = t;  //for next nodes
                last = t;
                last->next = first;
        }

}

void CList :: Display()
{
        CNode *tmp = first;
        if(first == NULL)
        {
                cout<<"List is empty";
                return;
```

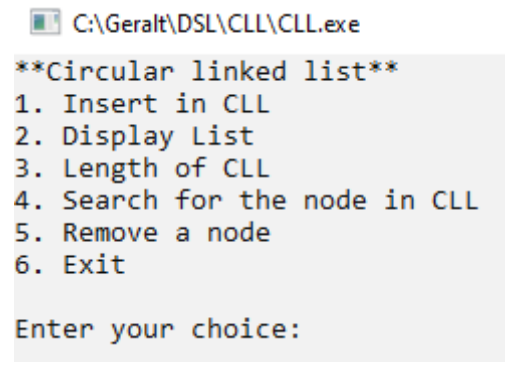
```
        }
        do
        {
                cout<<tmp->data<<" -> ";
                tmp = tmp->next;
        }
        while(tmp != first);
        cout<<" Back to first ";
}

void CList :: Length()
{
        int cnt=0;
        CNode *tmp = first;
        if(first == NULL)
        {
                cout<<"List is empty";
                return;
        }
        do
        {
                cnt++;
                tmp = tmp->next;
        }
        while(tmp != first);
        cout<<cnt;
}

void CList :: Search(int x)
{
        CNode *tmp = first;
        int flag = 0;
        if(first == NULL)
        {
                cout<<"List is empty";
                return;
        }
        do
        {
                if(tmp->data == x)
                {
                        flag = 1;
                        break;
                }
                tmp = tmp->next;
```

```
        }
        while(tmp != first);
        if(flag == 0)
        {
                cout<<x<<" not found ";
        }
        else
        {
                cout<<x<<" found ";
        }
}

void CList :: Remove(int x)
{
        CNode *tmp = first;
        CNode *prev = NULL;
        int flag = 0;
        if(first == NULL) //empty list and return control
        {
                cout<<"List is empty";
                return;
        }

        do
        {
                if(tmp->data == x)
                {
                        flag = 1;
                        break;
                }
                prev = tmp;
                tmp = tmp->next;

        }while(tmp != first);

        if(flag == 0) //unsuccessful deletion
        {
                cout<<x<<" not found";
                return;
        }

        if(first == last) //single node deletion
        {
                first == NULL;
                last == NULL;
```

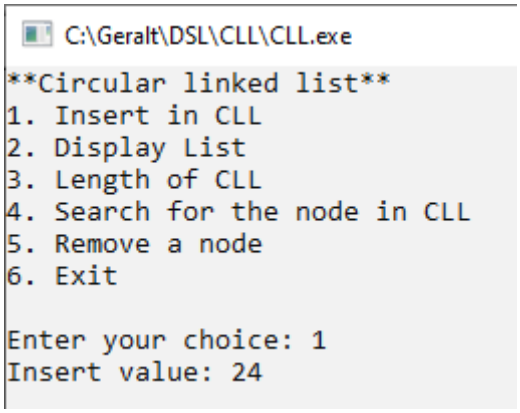
```
        }
        else if(tmp == first) //first node deletion
        {
                first = tmp->next;
                last->next = first;
        }
        else if(tmp == last) //last node deletion
        {
                last = prev;
                last->next = first;
        }
        else
        {
                prev->next = tmp->next;
        }
        delete tmp;
}


//Menu
int main()
{
        int ch;
        int num, y, z;
        CList c;
        while(1)
        {
                system("cls");

                cout<<"**Circular linked list**"<<endl;
                cout<<"1. Insert in CLL\n";
                cout<<"2. Display List\n";
                cout<<"3. Length of CLL\n";
                cout<<"4. Search for the node in CLL\n";
                cout<<"5. Remove a node\n";
                cout<<"6. Exit\n\n";

                cout<<"Enter your choice: ";
                cin>>ch;

                switch(ch)
                {
                        case 1:
                                        cout<<"Insert value: ";
                                        cin>>num;
```

```
                    c.Insert(num); //calling function
                    getch();
                    break;

            case 2:
                    cout<<"Display CLL: ";
                    c.Display();
                    getch();
                    break;

            case 3:
                    cout<<"Length of the list is: ";
                    c.Length();
                    getch();
                    break;

            case 4:
                    cout<<"Search the element: ";
                    cin>>y;
                    c.Search(y);
                    getch();
                    break;

            case 5:
                    cout<<"Enter element to be removed: ";
                    cin>>z;
                    c.Remove(z);
                    c.Display();
                    getch();
                    break;

            case 6:
                    cout<<"opt 6";
                    exit(1);
            default:
                    cout<<"Incorrect option bruv";
                    getch();
        }//end of switch
    }//end of while
}//end main
```

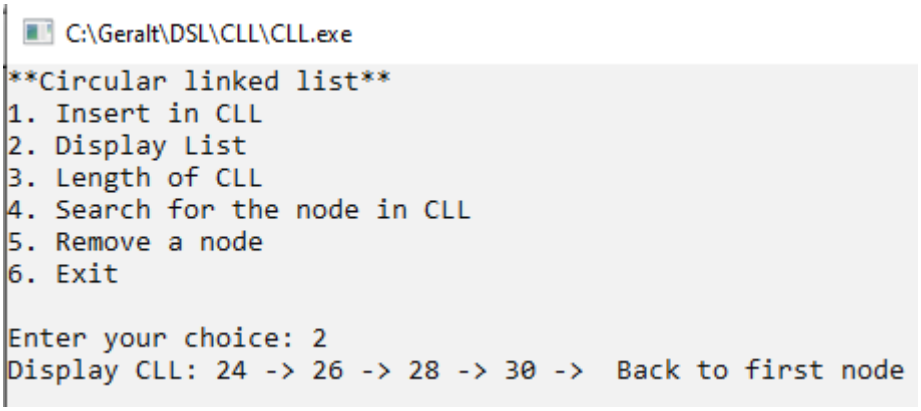
Output:



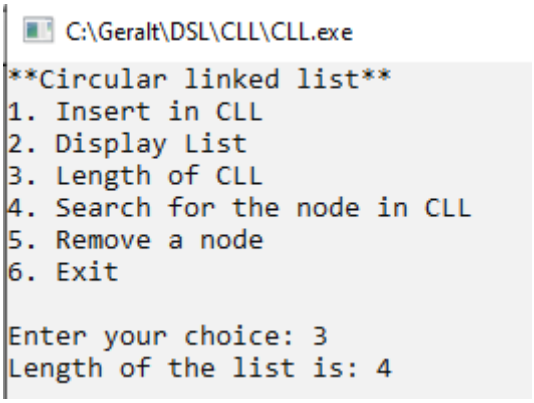
Enter elements:



Display List:



Display length:



Search element:
Element found:

```
C:\Geralt\DSL\CLL\CLL.exe
**Circular linked list**
1. Insert in CLL
2. Display List
3. Length of CLL
4. Search for the node in CLL
5. Remove a node
6. Exit

Enter your choice: 4
Search the element: 24
Element found
```

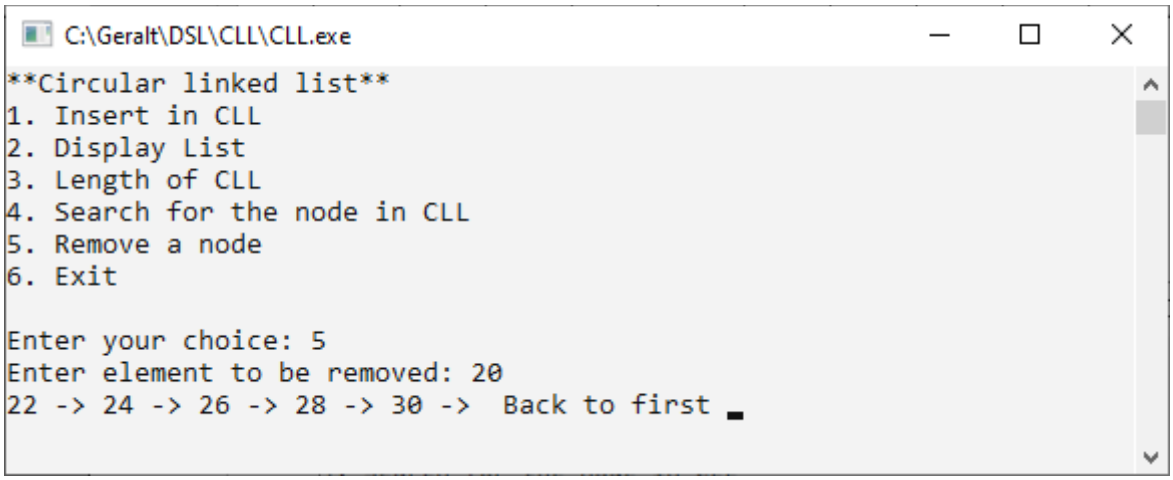
Element not found:

```
C:\Geralt\DSL\CLL\CLL.exe
**Circular linked list**
1. Insert in CLL
2. Display List
3. Length of CLL
4. Search for the node in CLL
5. Remove a node
6. Exit

Enter your choice: 4
Search the element: 12
Element not found
```

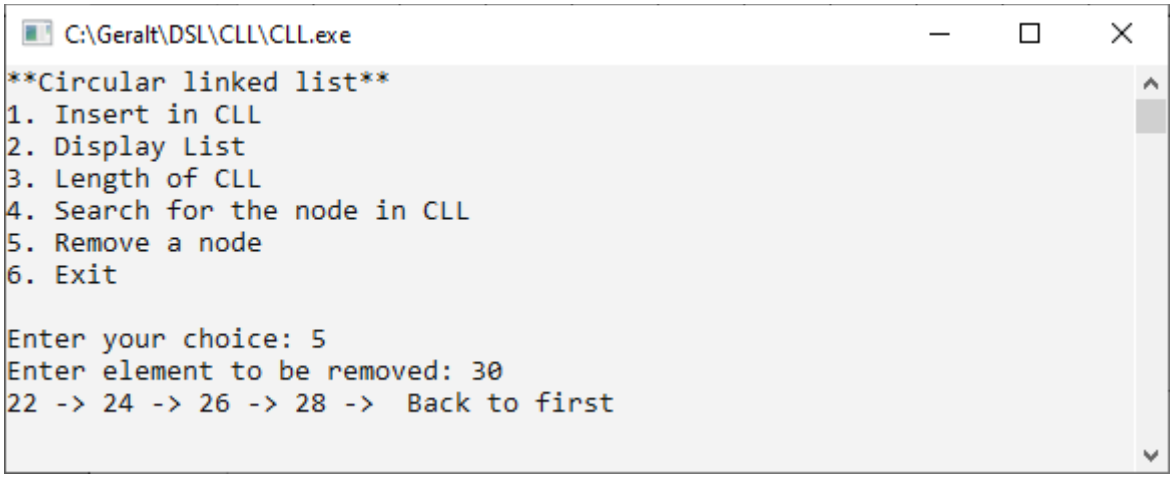
Remove a node:

```
C:\Geralt\DSL\CLL\CLL.exe
**Circular linked list**
1. Insert in CLL
2. Display List
3. Length of CLL
4. Search for the node in CLL
5. Remove a node
6. Exit

Enter your choice: 2
Display CLL: 20 -> 22 -> 24 -> 26 -> 28 -> 30 ->  Back to first
```

Head node deletion:



Tail node deletion:



Mid node deletion: