

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;

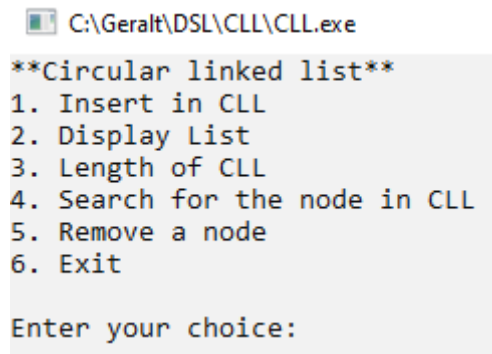
    case 6:
        cout<<"opt 6";
        exit(1);

    default:
        cout<<"Incorrect option bruv";
        getch();

    } //end of switch
} //end of while
} //end main

```

Output:



```

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:

```

Enter elements:







```
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: 1
Insert value: 24
```

Display List:

```
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: 24 -> 26 -> 28 -> 30 -> Back to first node
```

Display length:

```
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: 3
Length of the list is: 4
```

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:

```
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: 5
Enter element to be removed: 20
22 -> 24 -> 26 -> 28 -> 30 -> Back to first
```

Tail node deletion:

```
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: 5
Enter element to be removed: 30
22 -> 24 -> 26 -> 28 -> Back to first
```

Mid node deletion:

```
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: 5
Enter element to be removed: 26
22 -> 24 -> 28 -> Back to first
```