Data Structures

UCS301

Assignment-3

Submitted by:

Name: Bhavya Kakwani

Roll No: 101903365

Batch: 2CO14

<u>Q1.</u> Develop a menu driven program for the following operations of on a Circular as well as a Doubly Linked List.

(a) Insertion anywhere in the linked list (As a first node, as a last node, and after/before a specific node).

(b) Deletion of a specific node, say 'Delete Node 60'. That mean the node to be deleted may appear as a head node, last node or a node in between.

(c) Search for a node.

<u>Soln.</u>

(i) <u>Doubly Linked List</u>

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

struct node
{
    int data;
    node *next,*prev;
};

class doublyLinkedList
{
    node *head;
    public:
    doublyLinkedList()
    {
        head=NULL;
    }
    void insertBegining();
    void insertEnd();
    void insertBefore();
    void insertAfter();
    void deleteSpecific();
    void search();
    void display();
};
```

```
void doublyLinkedList::insertBegining()
{
    node *temp=new node;
    cout<<"Enter data : ";
    cin>>temp->data;
    temp->prev=temp->next=NULL;
    if(head==NULL)
        head=temp;
    else
    {
        temp->next=head;
        head->prev=temp;
        head=temp;
    }
}

void doublyLinkedList::insertEnd()
{
    node *temp=new node;
    cout<<"Enter data : ";
    cin>>temp->data;
    temp->prev=temp->next=NULL;
    if(head==NULL)
        head=temp;
    else
    {
        node *temp1=head;
        while(temp1->next!=NULL)
            temp1=temp1->next;
        temp1->next=temp;
        temp->prev=temp1;
    }
}
```

```
void doublyLinkedList::insertBefore()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        int x;
        node *temp1=head;
        cout<<"Enter the value before the element should be inserted : ";
        cin>>x;
        while(temp1!=NULL && temp1->data!=x)
            temp1=temp1->next;
        if(temp1==NULL)
            cout<<x<<" not found! \n";
        else
        {
            node *temp=new node;
            cout<<"Enter data : ";
            cin>>temp->data;
            temp->prev=temp->next=NULL;
            if(temp1==head)
            {
                temp->next=temp1;
                temp1->prev=temp;
                head=temp;
            }
            else
            {
                temp->next=temp1;
                temp->prev=temp1->prev;
                temp1->prev->next=temp;
                temp1->prev=temp;
            }
```

```
        }
    }
}

void doublyLinkedList::insertAfter()
{
    if(head==NULL)
        cout<<"List is empty \n";
    else
    {
        node *temp1=head;
        int x;
        cout<<"Enter the value after the element should be inserted : ";
        cin>>x;
        while(temp1!=NULL && temp1->data!=x)
            temp1=temp1->next;
        if(temp1==NULL)
            cout<<x<<" not found! \n";
        else
        {
            node *temp=new node;
            cout<<"Enter data : ";
            cin>>temp->data;
            temp->prev=temp->next=NULL;
            if(temp1->next==NULL)
            {
                temp->prev=temp1;
                temp1->next=temp;
            }
            else
            {
                temp->prev=temp1;
                temp->next=temp1->next;
                temp1->next->prev=temp;
```

```
                temp1->next=temp;
            }
        }
    }
}

void doublyLinkedList::deleteSpecific()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        node *temp1=head;
        int x;
        cout<<"Enter the element to delete : ";
        cin>>x;
        while(temp1!=NULL && temp1->data!=x)
            temp1=temp1->next;
        if(temp1==NULL)
            cout<<x<<" not found! \n";
        else
        {
            if(temp1==head)
            {
                temp1->next->prev=NULL;
                head=head->next;
            }
            else if(temp1->next==NULL)
                temp1->prev->next=NULL;
            else
            {
                temp1->prev->next=temp1->next;
                temp1->next->prev=temp1->prev;
            }
```

```
            delete temp1;
        }
    }
}

void doublyLinkedList::search()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        node *temp=head;
        int x,pos=0;
        cout<<"Enter the element to search : ";
        cin>>x;
        while(temp!=NULL && temp->data!=x)
        {
            temp=temp->next;
            pos++;
        }
        if(temp==NULL)
            cout<<x<<" not found! \n";
        else
            cout<<"Element found at position : "<<pos<<endl;
    }
}

void doublyLinkedList::display()
{
    if(head==NULL)
        cout<<"List is empty \n";
    else
    {
        node *temp=head;
```

```
        cout<<"List elements : \n";
        while(temp!=NULL)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
    }
}

int main()
{
    doublyLinkedList l;
    int ch;
    char c;
    do
    {
        cout<<"Menu of choices : \n";
        cout<<"1) Insert at begining \n";
        cout<<"2) Insert at end \n";
        cout<<"3) Insert before specific \n";
        cout<<"4) Insert after specific \n";
        cout<<"5) Delete specific \n";
        cout<<"6) Search \n";
        cout<<"7) Display \n";
        cout<<"8) Exit \n";
        cout<<"\nEnter your choice : ";
        cin>>ch;
        cout<<endl;
        switch(ch)
        {
            case 1:
                l.insertBegining();
                break;
            case 2:
```

```
                l.insertEnd();
                break;
            case 3:
                l.insertBefore();
                break;
            case 4:
                l.insertAfter();
                break;
            case 5:
                l.deleteSpecific();
                break;
            case 6:
                l.search();
                break;
            case 7:
                l.display();
                break;
            case 8:
                exit(0);
            default:
                cout<<"Invalid choice! \n";
        }
        cout<<"\n\nDo you want to continue? (y/n) : ";
        cin.ignore();
        cin>>c;
        system("cls");
    }while(c=='y'||c=='Y');
    return 0;
}
```

Output:

When choice is 1

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 1

Enter data : 2


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
2

Do you want to continue? (y/n) :
```

When choice is 2

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 2

Enter data : 3


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
2 3

Do you want to continue? (y/n) : y
```

When choice is 3

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 3

Enter the value before the element should be inserted : 2
Enter data : 8


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
8 2 3

Do you want to continue? (y/n) : y
```

When choice is 4

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 4

Enter the value after the element should be inserted : 8
Enter data : 5


Do you want to continue? (y/n) : y
```

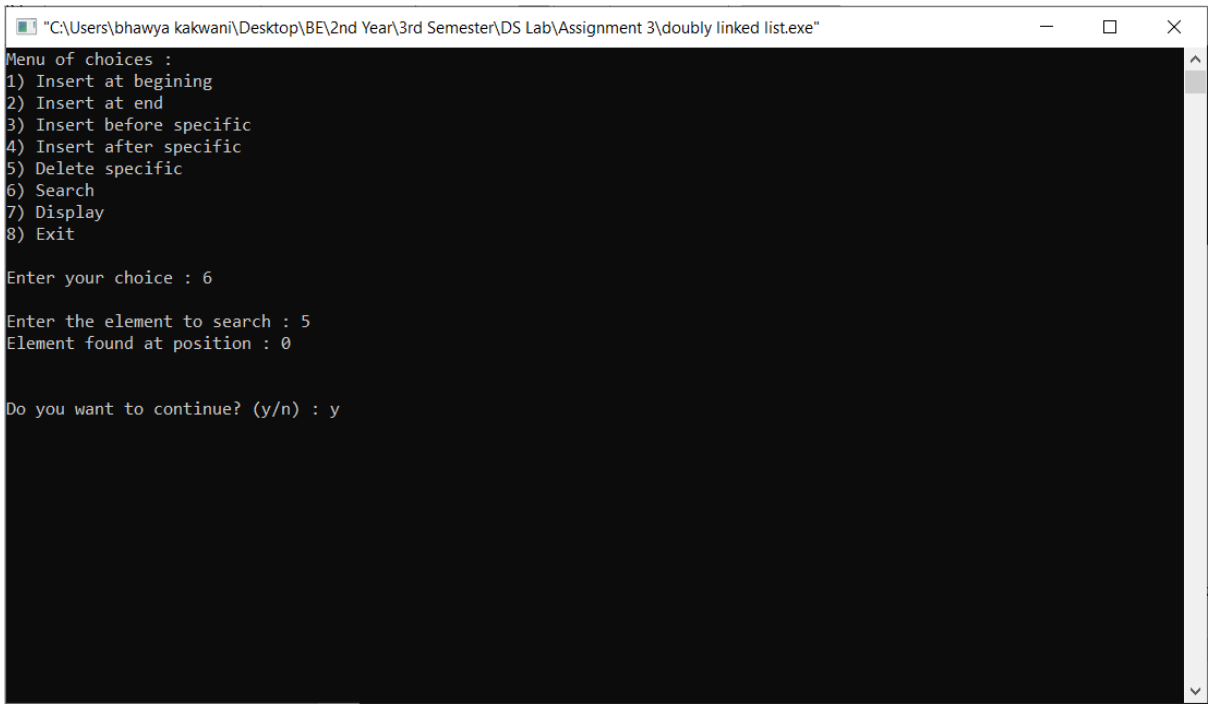
```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
8 5 2 3

Do you want to continue? (y/n) :
```

When choice is 5

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 5

Enter the element to delete : 8


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\doubly linked list.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
5 2 3

Do you want to continue? (y/n) :
```

When choice is 6



(ii) Singly Circular Linked List

```
#include<iostream>

using namespace std;


struct node

{

    int data;

    node *next;

};


class SinglyCircularLinkedList

{

    node *head;

    public:

    SinglyCircularLinkedList()

    {

        head=NULL;

    }

    void insertBegining();
```

```
    void insertEnd();
    void insertBefore();
    void insertAfter();
    void deleteSpecific();
    void search();
    void display();
};

void SinglyCircularLinkedList::insertBegining()
{
    node *temp=new node;
    cout<<"Enter data : ";
    cin>>temp->data;
    temp->next=temp;
    if(head==NULL)
        head=temp;
    else
    {
        node *temp1=head;
        while(temp1->next!=head)
            temp1=temp1->next;
        temp->next=head;
        temp1->next=temp;
        head=temp;
    }
}

void SinglyCircularLinkedList::insertEnd()
{
    node *temp=new node;
    cout<<"Enter data : ";
    cin>>temp->data;
    temp->next=temp;
    if(head==NULL)
```

```
        head=temp;
    else
    {
        node *temp1=head;
        while(temp1->next!=head)
            temp1=temp1->next;
        temp->next=head;
        temp1->next=temp;
    }
}

void SinglyCircularLinkedList::insertBefore()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        node *temp1=head,*pre=NULL;
        int x;
        cout<<"Enter the value before the element should be inserted : ";
        cin>>x;
        while(temp1->next!=head && temp1->data!=x)
        {
            pre=temp1;
            temp1=temp1->next;
        }
        if(temp1->data==x)
        {
            node *temp=new node;
            cout<<"Enter data : ";
            cin>>temp->data;
            if(temp1==head)
            {
                pre=head;
```

```
                while(pre->next!=head)
                    pre=pre->next;
                head=temp;
            }
            temp->next=temp1;
            pre->next=temp;
        }
        else
            cout<<x<<" not found! \n";
    }
}

void SinglyCircularLinkedList::insertAfter()
{
    if(head==NULL)
        cout<<"List is empty \n";
    else
    {
        node *temp1=head;
        int x;
        cout<<"Enter the value after the element should be inserted : ";
        cin>>x;
        while(temp1->next!=head && temp1->data!=x)
            temp1=temp1->next;
        if(temp1->data==x)
        {
            node *temp=new node;
            cout<<"Enter data : ";
            cin>>temp->data;
            temp->next=temp1->next;
            temp1->next=temp;
        }
        else
            cout<<x<<" not found in the list! \n";
```

```
    }
}

void SinglyCircularLinkedList::deleteSpecific()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        int x;
        node *temp=head,*prev=NULL;
        cout<<"Enter the element to delete : ";
        cin>>x;
        while(temp->next!=head && temp->data!=x)
        {
            prev=temp;
            temp=temp->next;
        }
        if(temp->data==x)
        {
            if(temp==head)
            {
                prev=head;
                while(prev->next!=head)
                    prev=prev->next;
                if(prev==head)
                {
                    head=NULL;
                    delete prev;
                }
                else
                {
                    prev->next=temp->next;
                    head=head->next;
```

```
                    delete temp;
                }
            }
            else
            {
                prev->next=temp->next;
                delete temp;
            }
        }
        else
            cout<<x<<" not found in the list! \n";
    }
}

void SinglyCircularLinkedList::search()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        int x,pos=0;
        node *temp=head;
        cout<<"Enter the element to search : ";
        cin>>x;
        while(temp->next!=head && temp->data!=x)
        {
            temp=temp->next;
            pos++;
        }
        if(temp->data==x)
            cout<<x<<" found at position : "<<pos<<endl;
    }
}
```

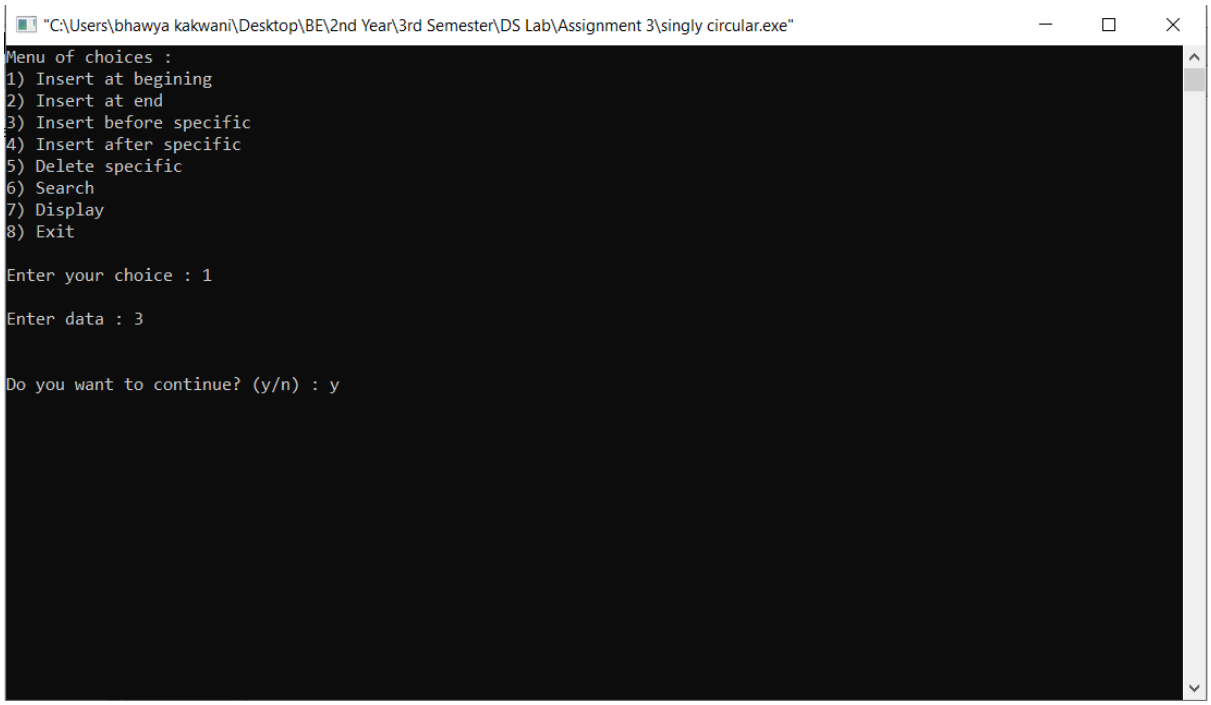
```
void SinglyCircularLinkedList::display()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        node *temp=head;
        cout<<"List elements : \n";
        while(temp->next!=head)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
        cout<<temp->data<<endl;
    }
}

int main()
{
    SinglyCircularLinkedList l;
    int ch;
    char c;
    do
    {
        cout<<"Menu of choices : \n";
        cout<<"1) Insert at begining \n";
        cout<<"2) Insert at end \n";
        cout<<"3) Insert before specific \n";
        cout<<"4) Insert after specific \n";
        cout<<"5) Delete specific \n";
        cout<<"6) Search \n";
        cout<<"7) Display \n";
        cout<<"8) Exit \n";
        cout<<"\nEnter your choice : ";
```

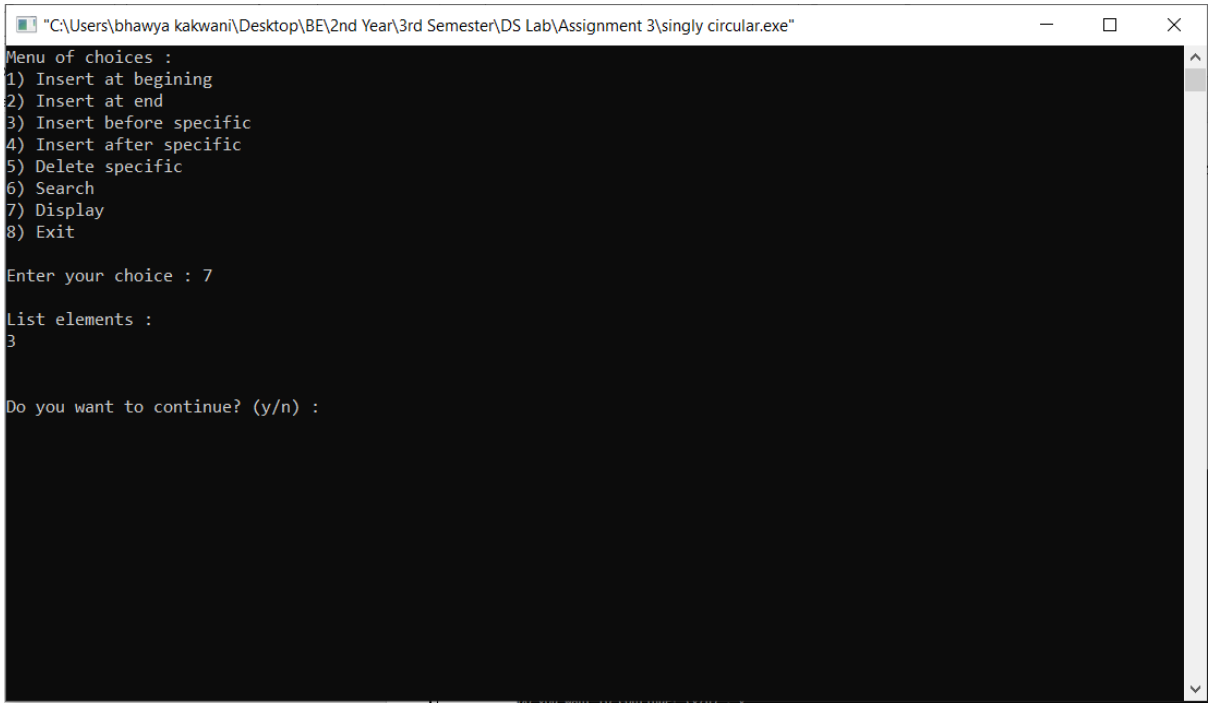
```
cin>>ch;
cout<<endl;
switch(ch)
{
    case 1:
        l.insertBegining();
        break;
    case 2:
        l.insertEnd();
        break;
    case 3:
        l.insertBefore();
        break;
    case 4:
        l.insertAfter();
        break;
    case 5:
        l.deleteSpecific();
        break;
    case 6:
        l.search();
        break;
    case 7:
        l.display();
        break;
    case 8:
        exit(0);
    default:
        cout<<"Invalid choice! \n";
}
cout<<"\n\nDo you want to continue? (y/n) : ";
cin.ignore();
cin>>c;
system("cls");
```

```
    }while(c=='y'||c=='Y');

    return 0;

}
```

Output:

When choice is 1





When choice is 2

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 2

Enter data : 2


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
3 2


Do you want to continue? (y/n) :
```

When choice is 3

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 3

Enter the value before the element should be inserted : 3
Enter data : 8


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
8 3 2


Do you want to continue? (y/n) :
```

When choice is 4

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 4

Enter the value after the element should be inserted : 8
Enter data : 5


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
8 5 3 2


Do you want to continue? (y/n) :
```

When choice is 5

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 5

Enter the element to delete : 8


Do you want to continue? (y/n) : y
```

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 7

List elements :
5 3 2


Do you want to continue? (y/n) :
```

When choice is 6

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\singly circular.exe"
Menu of choices :
1) Insert at begining
2) Insert at end
3) Insert before specific
4) Insert after specific
5) Delete specific
6) Search
7) Display
8) Exit

Enter your choice : 6

Enter the element to search : 3
3 found at position : 1


Do you want to continue? (y/n) : y
```

<u>Q2.</u> Display all the node values in a circular linked list, repeating value of head node at the end too. For example, if elements present in the circular linked list starting from head are 20 → 100 → 40 → 80 → 60, then output should be displayed as 20 100 40 80 60 20.

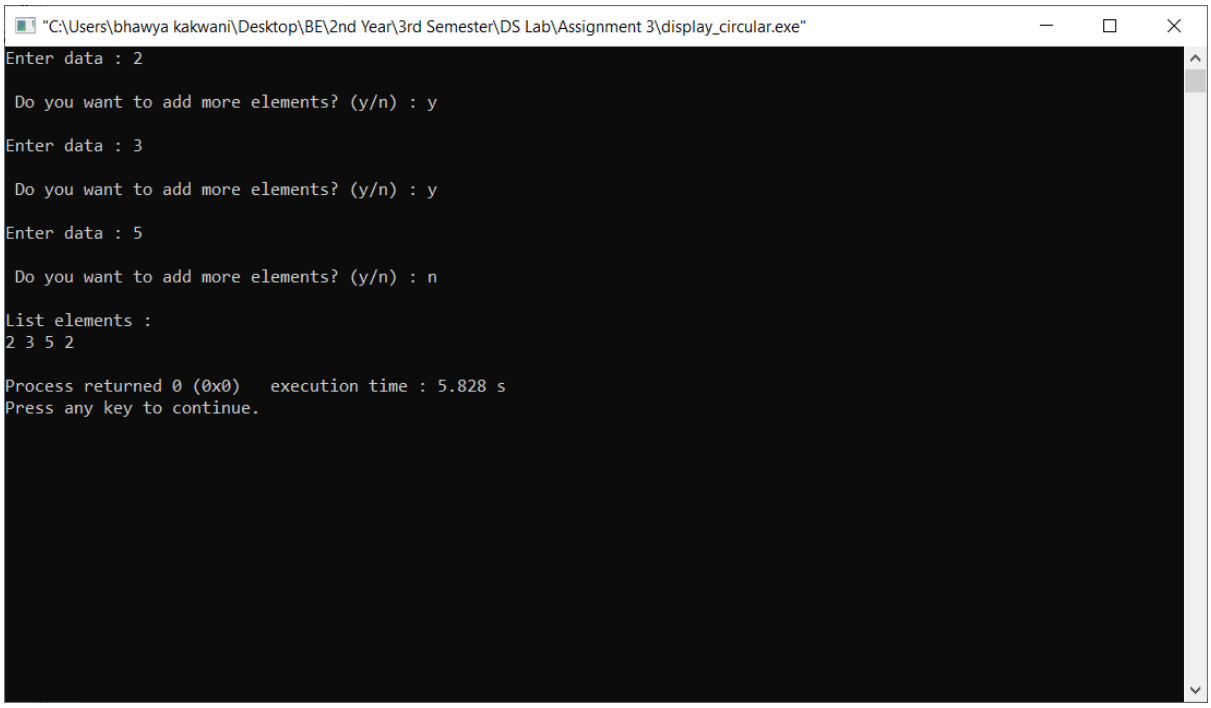
<u>Soln.</u>

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

struct node
{
    int data;
    node *next;
};

class SinglyCircularLinkedList
{
    node *head;
    public:
    SinglyCircularLinkedList()
    {
        head=NULL;
    }
    void create();
    void display();
};

void SinglyCircularLinkedList::create()
{
    char ch;
    do
    {
        node *temp=new node;
        cout<<"Enter data : ";
        cin>>temp->data;
        temp->next=temp;
        if(head==NULL)
```

```
            head=temp;
        else
        {
            node *temp1=head;
            while(temp1->next!=head)
                temp1=temp1->next;
            temp->next=head;
            temp1->next=temp;
        }
        cout<<"\n Do you want to add more elements? (y/n) : ";
        cin>>ch;
        cout<<endl;
    }while(ch=='y'||ch=='Y');
}

void SinglyCircularLinkedList::display()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        node *temp=head;
        cout<<"List elements : \n";
        while(temp->next!=head)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
        cout<<temp->data<<" ";
        temp=temp->next;
        cout<<temp->data<<endl;
    }
}
```

```
int main()
{
    SinglyCircularLinkedList l;
    l.create();
    l.display();
    return 0;
}
```

Output:



<u>Q3.</u> Write a program to find size of

(a) Doubly Linked List.

(b) Circular Linked List.

<u>Soln.</u>

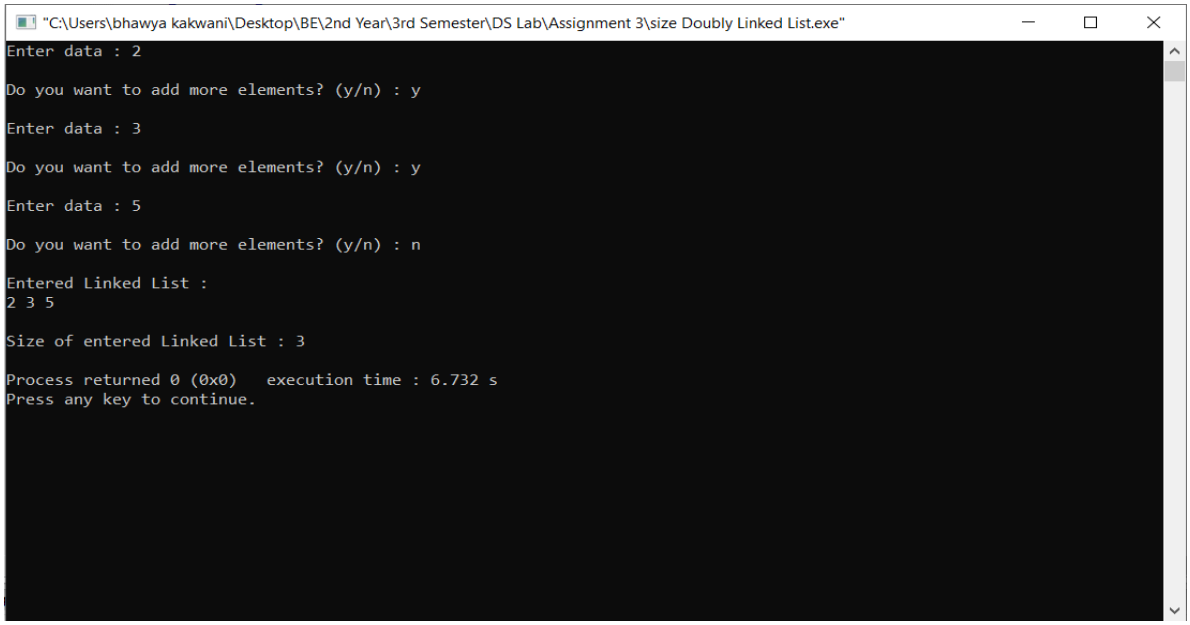
(a) Doubly Linked List

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

struct node
{
    int data;
    node *next,*prev;
};

class doublyLinkedList
{
    node *head;
    public:
    doublyLinkedList()
    {
        head=NULL;
    }
    void create();
    void display();
    void calculateSize();
};

void doublyLinkedList::create()
{
    char ch;
    do
    {
        node *temp=new node;
        cout<<"Enter data : ";
```

```
        cin>>temp->data;
        temp->prev=temp->next=NULL;
        if(head==NULL)
            head=temp;
        else
        {
            node *temp1=head;
            while(temp1->next!=NULL)
                temp1=temp1->next;
            temp1->next=temp;
            temp->prev=temp1;
        }
        cout<<"\nDo you want to add more elements? (y/n) : ";
        cin>>ch;
        cout<<endl;
    }while(ch=='y'||ch=='Y');
}

void doublyLinkedList::display()
{
    if(head==NULL)
        cout<<"List is empty \n";
    else
    {
        node *temp=head;
        cout<<"Entered Linked List : \n";
        while(temp!=NULL)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
        cout<<endl;
    }
}
```

```
void doublyLinkedList::calculateSize()
{
    node *temp=head;
    int size=0;
    while(temp!=NULL)
    {
        temp=temp->next;
        size++;
    }
    cout<<"Size of entered Linked List : "<<size<<endl;
}

int main()
{
    doublyLinkedList l;
    l.create();
    l.display();
    cout<<endl;
    l.calculateSize();
    return 0;
}
```

Output:



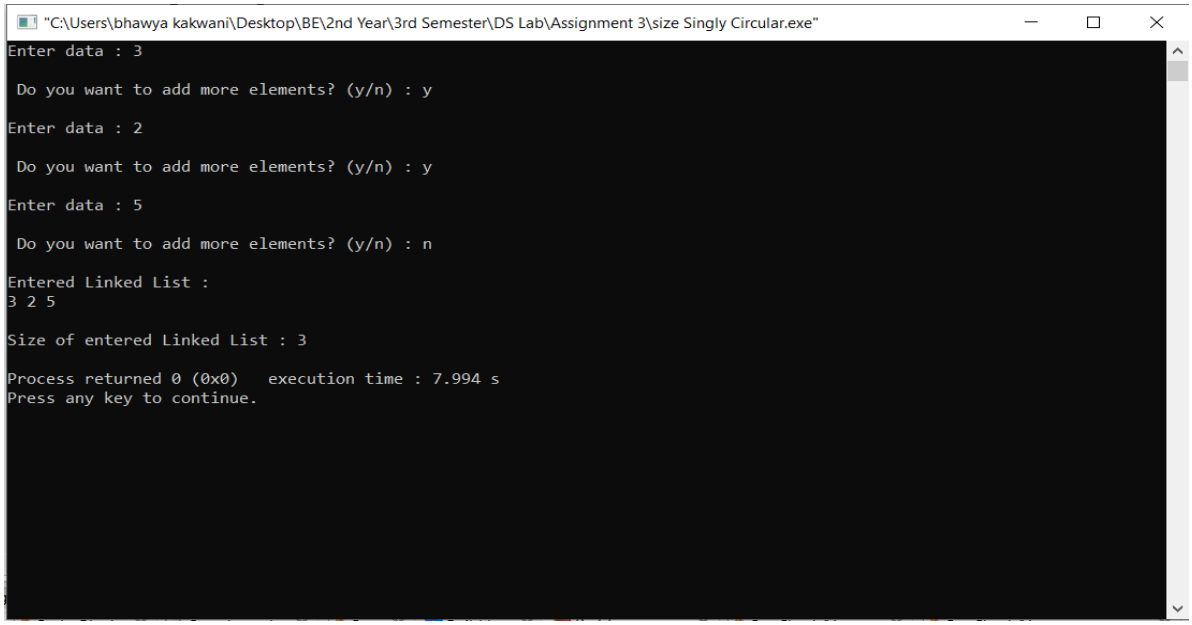
(b) Circular Linked List

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

struct node
{
    int data;
    node *next;
};

class SinglyCircularLinkedList
{
    node *head;
    public:
    SinglyCircularLinkedList()
    {
        head=NULL;
    }
    void create();
    void display();
    void calculateSize();
};

void SinglyCircularLinkedList::create()
{
    char ch;
    do
    {
        node *temp=new node;
        cout<<"Enter data : ";
        cin>>temp->data;
        temp->next=temp;
        if(head==NULL)
            head=temp;
```

```
        else
        {
            node *temp1=head;
            while(temp1->next!=head)
                temp1=temp1->next;
            temp->next=head;
            temp1->next=temp;
        }
        cout<<"\n Do you want to add more elements? (y/n) : ";
        cin>>ch;
        cout<<endl;
    }while(ch=='y'||ch=='Y');
}

void SinglyCircularLinkedList::display()
{
    if(head==NULL)
        cout<<"List is empty! \n";
    else
    {
        node *temp=head;
        cout<<"Entered Linked List : \n";
        while(temp->next!=head)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
        cout<<temp->data<<endl;
    }
}

void SinglyCircularLinkedList::calculateSize()
{
    int size=0;
```

```
    if(head!=NULL)
    {
        node *temp=head;
        while(temp->next!=head)
        {
            temp=temp->next;
            size++;
        }
        size++;
    }
    cout<<"Size of entered Linked List : "<<size<<endl;
}

int main()
{
    SinglyCircularLinkedList l;
    l.create();
    l.display();
    cout<<endl;
    l.calculateSize();
    return 0;
}
```

Output:



<u>Q4.</u> Write a program to check if a doubly linked list of characters is palindrome or not.

<u>Soln.</u>

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

struct node
{
    char data;
    node *next,*prev;
};

class doublyLinkedList
{
    node *head;
    public:
    doublyLinkedList()
    {
        head=NULL;
    }
    void create();
    void display();
    void checkPalindrome();
};

void doublyLinkedList::create()
{
    char ch;
    do
    {
        node *temp=new node;
        cout<<"Enter data : ";
        cin>>temp->data;
        temp->prev=temp->next=NULL;
        if(head==NULL)
```

```
            head=temp;
        else
        {
            node *temp1=head;
            while(temp1->next!=NULL)
                temp1=temp1->next;
            temp1->next=temp;
            temp->prev=temp1;
        }
        cout<<"\nDo you want to add more elements? (y/n) : ";
        cin>>ch;
        cout<<endl;
    }while(ch=='y'||ch=='Y');
}

void doublyLinkedList::display()
{
    if(head==NULL)
        cout<<"List is empty \n";
    else
    {
        node *temp=head;
        cout<<"Entered Linked List : \n";
        while(temp!=NULL)
        {
            cout<<temp->data;
            temp=temp->next;
        }
        cout<<endl;
    }
}

void doublyLinkedList::checkPalindrome()
{
```

```
    if(head==NULL)
        cout<<"Linked list is empty! \n";
    else
    {
        int flag=0;
        node *temp,*tail;
        temp=tail=head;
        while(tail->next!=NULL)
            tail=tail->next;
        while(temp!=tail)
        {
            if(temp->data!=tail->data)
            {
                flag=1;
                cout<<"Entered list is not a palindrome! \n";
                break;
            }
            temp=temp->next;
            tail=tail->prev;
        }
        if(flag==0)
            cout<<"Entered list is a palindrome! \n";
    }
}

int main()
{
    doublyLinkedList l;
    l.create();
    l.display();
    cout<<endl;
    l.checkPalindrome();
    return 0;
}
```

Output:

```
"C:\Users\bhawya kakwani\Desktop\BE\2nd Year\3rd Semester\DS Lab\Assignment 3\Check Palindrome.exe"
Enter data : p

Do you want to add more elements? (y/n) : y

Enter data : o

Do you want to add more elements? (y/n) : y

Enter data : p

Do you want to add more elements? (y/n) : n

Entered Linked List :
pop

Entered list is a palindrome!

Process returned 0 (0x0)   execution time : 75.474 s
Press any key to continue.
```

<u>Q5.</u> Write a program to check if a linked list is Circular Linked List or not.

<u>Soln.</u>

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

struct node
{
    int data;
    node *next;
};

bool isCircular(node *head)
{
    if(head==NULL)
       return true;
    node *temp = head;
    while(temp->next!=NULL && temp->next!=head)
       temp=temp->next;
    if(temp->next==head)
        return true;
    else
        return false;
}

node *newNode(int data)
{
    node *temp=new node;
    temp->data=data;
    temp->next=NULL;
    return temp;
}

int main()
{
```

```
    // Start with the empty list
    node *head=newNode(41);
    head->next=newNode(17);
    head->next->next=newNode(8);
    if(isCircular(head))
        cout<<"Yes, Linked list is circular! \n";
    else
        cout<<"No, Linked list is not circular! \n";
    // Making linked list circular
    head->next->next->next=head;
    if(isCircular(head))
        cout<<"Yes, Linked list is circular! \n";
    else
        cout<<"No, Linked list is not circular! \n";
    return 0;
}
```

Output: