TASK-07

Create a menu driven program for Circular Linked List.
Include all operations in the menu, which are as follows:
1. Insert at Head
2. Insert at Last
3. Insert After
4. Insert Before
5. Delete from Head
6. Delete Node
7. Traverse List

Code:

- **Cnode.h File**

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

class Cnode
{
  public:
        double data;
        Cnode* next;
        Cnode(double i=0, Cnode* n=0)
                {
                        data = i;
                        next = n;
                }
};
```

- **Clinkedlist.h file**

```
#include"Cnode.h"
#include<iostream>
using namespace std;
```

```
class Clinkedlist
{
        private:
                Cnode* Head;
        public:
           CLinkedList()
                {
                        Head = 0;
                }

                void insertathead( double value )
                {
                        Cnode* newNode = new Cnode( value );
                        if ( Head == 0 )
                        {
                                Head = newNode;
                                newNode->next = Head;
                        }

                        else
                        {
                                Cnode* current = Head;
                                while ( current->next!= Head )
                                {
                                        current = current->next;
                                }

                                 Head = newNode;
                                 current->next = newNode;
                        }
                }

                void insertatlast( double value )
                {
```

```
                                        Cnode* newNode = new Cnode( value );
                                        if ( Head == 0 )
                                        {
                                                Head = newNode;
                                                newNode->next = Head;
                                        }

                                        else
                                        {
                                                Cnode* current = Head;
                                                while ( current->next!= Head )
                                                {
                                                        current = current->next;
                                                }
                                                newNode->next = current->next;
                                                current->next = newNode;
                                        }
                                }

                                void insertafter( double existing , double value )
                {
                                if(Head == 0)
                                {
                                        cout<<"\nList is empty.";
                                }
                                else
                                {
                                        Cnode* currnode = Head;
                                        while(currnode   !=   0   &&   currnode->data   !=
                existing)
                                        {
                                                currnode = currnode->next;
                                        }
                                        if(currnode==0)
```

```
                    {
                            cout<<"\nInsertion is not possible in the list
because existing element in not present in the list.";
                    }
                    else
                    {
                            Cnode* newnode = new Cnode(value);
                            newnode->next = currnode->next;
                            currnode->next = newnode;
                    }
            }
    }

  void insertbefore( double existing , double value )
  {
            if(Head == 0)
            {
                    cout<<"\nList is empty.";
            }
            else if(existing == Head->data)
            {
                    insertathead(value);
            }
            else
            {
                    Cnode* prevnode = 0;
                    Cnode* currnode = Head;
                    while(currnode   !=   0   &&   currnode->data   !=
existing)
                    {
                            prevnode = currnode;
                            currnode = currnode->next;
                    }
                    if(currnode==0)
```

```
                        {
                                cout<<"\nInsertion is not possible in the list
because existing element in not present in the list.";
                        }
                        else
                        {
                                Cnode* newnode = new Cnode(value);
                                newnode->next = currnode;        // newnode
= currnode
                                prevnode->next = newnode;        // currnode-
>next = currnode
                        }
                }
        }

        void deletefromhead()
        {
                if ( Head == 0 )
                {
                        cout<<" List is empty. "<<endl;
                }

                else
                {
                        Cnode* delNode = Head;
                        Cnode* current = Head;
                        while( current->next!= Head )
                        {
                                current = current->next;
                        }
                        current->next = Head->next;
                        Head = Head->next;
                        delNode->next = 0;
                        delete delNode;
                }
        }

        void deletespecific( double existing )
        {
                if ( Head == 0 )
                {
                        cout<<" List is empty. "<<endl;
                }
                else if ( existing == Head->data)
                {
                        deletefromhead();
                }
                else
                {
                        Cnode* current = Head->next;
                        Cnode* prev = Head;
                        while ( current!=Head && current->data!=existing)
                        {
                                prev = current;
                                current = current->next;
                        }
                        if ( current == Head )
                        {
                                cout<<" value not existing. "<<endl;
                        }
                        else
                        {
                                prev->next = current->next;
                                current->next = 0;
                                delete current;
                        }
                }
        }
```

```
        void traverselist()
        {
                if(Head == 0)
                {
                        cout<<"\nList is empty.";
                }
                else
                {
                        cout<<"\nValues in list are: "<<endl;
                        Cnode* currnode = Head;
                        while(currnode != Head)
                        {
                                cout<<currnode->data<<endl;
                                currnode = currnode->next;
                        }
                }
        }
};
```

- **.cpp file**

```
#include <iostream>
#include "Clinkedlist.h"
using namespace std;
int main()
{
        double value;
        double existing;
        char con;
        int choice;
        Clinkedlist list;
        do
        {
```

```
        cout<<"\tPress 1 for insert at head"<<endl;
        cout<<"\tPress 2 for insert at last"<<endl;
        cout<<"\tPress 3 for insert after"<<endl;
        cout<<"\tPress 4 for insert before"<<endl;
        cout<<"\tPress 5 for delete from head"<<endl;
        cout<<"\tPress 6 for delete from specific node"<<endl;
        cout<<"\tPress 7 for traverse node"<<endl;
        cout<<"Enter choice: ";
        cin>>choice;
        switch (choice)
        {
                case 1:
                cout<<"Enter value to insert at head: ";
                cin>>value;
                list.insertathead(value);
                break;
    case 2:
                cout<<"Enter value to insert at tail: ";
                cin>>value;
                list.insertatlast(value);
           break;
        case 3:
                cout<<"Enter value to insert after: ";
                cin>>existing;
                cin>>value;
                list.insertafter(existing,value);
            break;
   case 4:
                cout<<"Enter value to insert before: ";
                cin>>existing;
                cin>>value;
                list.insertbefore(existing,value);
           break;
```

```
                    case 5:
                            list.deletefromhead();
                       break;
                    case 6:
                            cout<<"Enter value for specific node deletion: ";
                            cin>>value;
                            list.deletespecific(value);
                            break;
                    case 7:
                            list.traverselist();
                       break;
                    default:
                            cout<<"Sorry! Wrong choise"<<endl;
                            break;
                    }
                    cout<<"\nPress (y) for again continue the program and
press any key except (y) for exit: ";
                    cin>>con;

            }
            while(con == 'y');
}
```