Circular Linked list

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

class Node{

    friend class DLL;
    public:
        int data;
        Node* next;
        Node* prev;

        Node(int val){
            data = val;
            next = NULL;
            prev = NULL;
        }
};

class DLL{
    public:
        Node* head = NULL;
        Node* tail = NULL;
        int size = 0;

        void insertAtHead(int val){

            size++;
            Node* newNode = new Node(val);

            if(head == NULL){
                head = newNode;
                tail = newNode;
            }
            else{
                newNode->next = head;
                head->prev = newNode;
                head = newNode;
            }

            cout<<"\nNew List is: ";
            traverse();
        }

        void delete2(){
            size--;
            Node* temp = head;
```

```
        if(temp == NULL){
            cout<<"\nLinked list is empty!!"<<endl;
            return;
        }
        if(temp->next == NULL){
            cout<<"\nLinked list contains only 1 element!!"<<endl;
            return;
        }

        temp = temp->next;

        if(temp == tail){
            (temp->prev)->next = NULL;
            tail = temp->prev;
            delete temp;
        }
        else{
            (temp->prev)->next = temp->next;
            (temp->next)->prev = temp->prev;
            delete temp;
        }

        cout<<"\nNew List is: ";
        traverse();
    }

    void traverse(){
        Node* temp = head;
        while(temp != NULL){
            cout<<temp->data<<" ";
            temp = temp->next;
        }
    }
    void displayODD(){
        Node* temp = head;
        if(temp == NULL){
            cout<<"\nLinked list is empty!!"<<endl;
        }
        while(temp != NULL){
            cout<<temp->data<<" ";
            if(temp->next != NULL)
                temp = temp->next->next;
            else break;
        }
    }

    double average(){
```

```
            Node* temp = head;
            int sum = 0;
            int count = 0;
            while(temp != NULL){
                sum += temp->data;
                count++;
                temp = temp->next;
            }

            double avg = double(sum)/count ;

            return avg;
        }

        void middleElement(){
            int count = 0;

            Node* temp = head;

            if(size == 0){
                cout<<"\nLinked list is empty!!"<<endl;
                return;
            }
            if(size %2 != 0){
                count = (size+1)/2;
                for(int i = 0; i<count-1; i++)
                    temp= temp->next;
                cout<<"\nMiddle element is: "<<temp->data<<endl;
            }
            if(size %2 == 0){
                count = size/2;
                for(int i = 0; i<count-1; i++)
                    temp= temp->next;
                cout<<"\nMiddle element are: "<<temp->data<<" , "<<temp->next-
>data<<endl;
            }
        }

        bool areEqual(DLL l){
            Node* temp1 = head;
            Node* temp2 = l.head;

            if(size != l.size)
                return false;

            while(temp1 != NULL && temp2 != NULL){
                if(temp1->data != temp2->data){
                    return false;
```

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

            return true;
        }
};


int main(){
    DLL l, l2;
    int choice, val, ch, check;
    cout<<"##MENU: \n1. Insert at Head \n2. DeleteAt2 \n3. Display ODD element
s \n4. Average \n5. Middle Element \n6. Compare linked lists \n";
    do{
        cout<<"\n\nEnter your choice: ";
        cin>>choice;
        switch(choice){
            case 1: cout<<"\nEnter value: ";
                    cin>>val;
                    l.insertAtHead(val);
                    break;
            case 2: l.delete2();
                    break;
            case 3: l.displayODD();
                    break;
            case 4: cout<<"\nAverage of the elements in linked list is = "<<l.
average()<<endl;
                    break;
            case 5: l.middleElement();
                    break;
            case 6: cout<<"\nFirst create a new linked list to compare with pr
evious =>"<<endl<<"Add elements to Linked List 2: ";
                    do{
                        cout<<"\nEnter value: ";
                        cin>>val;
                        l2.insertAtHead(val);
                        cout<<"\nEnter 0 to stop: ";
                        cin>>check;
                    }while(check != 0);
                    cout<<"\nResult of comaparison: "<<l.areEqual(l2);
                    break;

            default: cout<<"\nWrong choice entered! \n";
        }
        cout<<"\nEnter 1 to continue and other to exit: ";
        cin>>ch;
    }while(ch == 1);
```

```
}
```