Question5.cpp

```cpp
// <----Lab 04 - Doubly and Circular Linked List---->

// Q5. Break the above-created circular linked list into two halves.

#include<iostream>
using namespace std;

class node {
    public:
        int data;
        node* next;
        node(int value) {
            data=value;
            next=NULL;
        }
        node(int value,node* nxt) {
            data=value;
            next=nxt;
        }
};

class CLL{
    node* head=NULL;
    int nodecount=0;
    public:
    void setlist(node* h,int n){
        head=h;
        nodecount=n;
    }
    void makehalves(CLL &obj){
        int middleposition = (int)nodecount/2;     //Amount for 1st half
        int secondhalf = nodecount-middleposition;  //Amount for 2nd half
        int count=0;
        node* temp=head;
        node* prev=temp;
        while(temp->next!=head&&count<middleposition) {  // prev is last element of first half, temp is head of second half
            prev=temp;
            temp=temp->next;
            count++;
        }
        prev->next=head; //detach prev from remaining half, circular link back to start
        node* secondhead=temp;
        while(temp->next!=head){
            temp=temp->next;
        }
        temp->next=secondhead;
        obj.setlist(secondhead,secondhalf);
    }

    void appendNode(int value){ //insert at end of list / tail
        if(head==NULL) {  //Check if Linked List Empty.
            node* n=new node(value,n);
```

```
53              head=n;
54              head->next=head;
55              nodecount++;
56              return;
57          }
58          else{
59              node* temp=head;
60              while(temp->next!=head) {
61                  temp=temp->next;
62              }
63              node* n=new node(value,head);
64              temp->next=n;
65              nodecount++;
66          }
67      }
68      void prependNode(int value){ //insert at start of list / head
69          if(head==NULL) {  //Check if Linked List Empty.
70              node* n=new node(value,n);
71              head=n;
72              head->next=head;
73              nodecount++;
74              return;
75          }
76          node* n=new node(value,head);
77          node* temp=head;
78          while(temp->next!=head) {
79              temp=temp->next;
80          }
81          temp->next=n;
82          head=n;
83          nodecount++;
84      }
85      void insertNodeAfter(int posvalue,int value){  // insert at position (i+1)
86          if(head==NULL) {  //Check if Linked List Empty.
87              cout<<"Empty List, adding at Head.\n";
88              node* n=new node(value,n);
89              head=n;
90              nodecount++;
91              return;
92          }
93          if(posvalue>nodecount-1){
94              cout<<"Position more than nodes in list, Inserting at tail.\n";
95              appendNode(value);
96              return;
97          }
98          int count=0;
99          node* temp=head;
100         while(temp->next!=head&&count<posvalue) {
101             temp=temp->next;
102             count++;
103         }
104         node* n=new node(value,temp->next);
105         temp->next=n;
106     }
107     void deleteathead(){
108         node* temp=head;
```

```
109         while(temp->next=head){
110             temp=temp->next;
111         }
112         temp->next=head->next;
113     }
114     void deleteattail(){
115         node* temp=head;
116         while(temp->next->next=head){
117             temp=temp->next;
118         }
119         temp->next=head;
120     }
121     void deleteNodeByKey(int value){  // delete by value
122         node* temp=head;
123         node* prev=temp;
124         while(temp->next!=head&&temp->data!=value) {
125             prev=temp;
126             temp=temp->next;
127         }
128         if(temp->data==value){
129             prev->next=temp->next;  //skip temp (i.e delete)
130             delete temp;
131         }
132         if(temp->next==head){
133             cout<<"Value not in Linked List.\n";
134             return;
135         }
136     }
137     void updateNodeByKey(int value){  // update by value
138         node* temp=head;
139         while(temp->next!=head&&temp->data!=value) {
140             node* prev=temp;
141             temp=temp->next;
142         }
143         if(temp->data==value){
144             cout<<"Enter new value: ";
145             cin>>temp->data;
146         }
147         if(temp->next==head){
148             cout<<"Value not in Linked List.\n";
149             return;
150         }
151     }
152     void print(){
153         node* temp=head;
154         cout<<"[HEAD] ";
155         if(head!=NULL){
156         cout<<temp->data<<"| "<<temp->next<<" -> ";
157         temp=temp->next;
158         while(temp!=head) {
159             cout<<temp->data<<"| "<<temp->next<<" -> ";
160             temp=temp->next;
161         }
162         cout<<"head [TAIL]"<<endl;
163         }
164         else{
```

```
165             cout<<"NULL [TAIL]"<<endl;
166         }
167     }
168 };
169 
170 int main(){
171     CLL list,list2;
172     float input=0;
173     int value;
174     while(input!=0.5) {
175         cout<<"-----------------------------------------------------\n";
176         cout<<"CURRENT LINKED LIST:\n";
177         list.print();
178         cout<<"-----------------------------------------------------\n";
179         cout<<"What would you like to do with the linked list?\n";
180         cout<<"1. Insert\t2. Delete\t3. Update\t4. Halve the Linked List [Halving will end
    the program]\nEnter 0.5 to Exit\n[Anything else will default to Delete]\n";
181         cin>>input;
182         if(input==1) {
183             cout<<"Enter Value to insert: ";
184             cin>>value;
185             cout<<"Where to Insert in Linked List?\n";
186             cout<<"1. At head\t2. At tail\t3. At specified Position\n[Any other value will
    default to Insertion at Head]\n";
187             cin>>input;
188             if(input == 2){
189                 list.appendNode(value);
190             }
191             else if(input == 3){
192                 int pos;
193                 cout<<"Enter the Position to insert After: ";
194                 cin>>pos;
195                 list.insertNodeAfter(pos,value);
196             }
197             else{
198                 list.prependNode(value);
199             }
200 
201         }
202         else if(input==0.5){
203             break;
204         }
205         else if(input==3){
206             cout<<"Enter the Value to Update: ";
207             cin>>value;
208             list.updateNodeByKey(value);
209         }
210         else if(input==4){
211             cout<<"Halving list"<<endl;
212             list.makehalves(list2);
213             cout<<"First Half list--------------------------------"<<endl;
214             list.print();
215             cout<<"Second Half list-------------------------------"<<endl;
216             list2.print();
217             return 0;
218         }
219         else{
```

```
            cout<<"Where to Delete from Linked List?\n";
            cout<<"1. At head\t2. At tail\t3. Delete a specific Value\n[Any other value will default to Deletion from Head]\n";
            cin>>input;
            if(input == 2){
                list.deleteattail();
            }
            else if(input == 3){
                int pos;
                cout<<"Enter the Value to Delete: ";
                cin>>value;
                list.deleteNodeByKey(value);
            }
            else{
                list.deleteathead();
            }
        }

    }
}
```