Question5.cpp

```cpp
// <---Lab 04 - Doubly and Circular Linked List--->
/*5. 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.
```

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

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

```
        else{
            cout<<"NULL [TAIL]"<<endl;
        }
    }
};

int main(){
    CLL list,list2;
    float input=0;
    int value;
    while(input!=0.5) {
        cout<<"-----------------------------------------------------\n";
        cout<<"CURRENT LINKED LIST:\n";
        list.print();
        cout<<"-----------------------------------------------------\n";
        cout<<"What would you like to do with the linked list?\n";
        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";
        cin>>input;
        if(input==1) {
            cout<<"Enter Value to insert: ";
            cin>>value;
            cout<<"Where to Insert in Linked List?\n";
            cout<<"1. At head\t2. At tail\t3. At specified Position\n[Any other value will default to Insertion at Head]\n";
            cin>>input;
            if(input == 2){
                list.appendNode(value);
            }
            else if(input == 3){
                int pos;
                cout<<"Enter the Position to insert After: ";
                cin>>pos;
                list.insertNodeAfter(pos,value);
            }
            else{
                list.prependNode(value);
            }

        }
        else if(input==0.5){
            break;
        }
        else if(input==3){
            cout<<"Enter the Value to Update: ";
            cin>>value;
            list.updateNodeByKey(value);
        }
        else if(input==4){
            cout<<"Halving list"<<endl;
            list.makehalves(list2);
            cout<<"First Half list--------------------------------"<<endl;
            list.print();
            cout<<"Second Half list-------------------------------"<<endl;
            list2.print();
            return 0;
        }
```

```
220         else{
221             cout<<"Where to Delete from Linked List?\n";
222             cout<<"1. At head\t2. At tail\t3. Delete a specific Value\n[Any other value will
    default to Deletion from Head]\n";
223             cin>>input;
224             if(input == 2){
225                 list.deleteattail();
226             }
227             else if(input == 3){
228                 int pos;
229                 cout<<"Enter the Value to Delete: ";
230                 cin>>value;
231                 list.deleteNodeByKey(value);
232             }
233             else{
234                 list.deleteathead();
235             }
236         }
237 
238     }
239 }
```