Question3.cpp

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

// Q3. Using the above created list N, sort the contents of list N is descending order.

#include<iostream>
using namespace std;

class node {
    public:
        int data;
        node* nextnode;   //to point to node after it
        node* prevnode;  //to point to node before it
        node() {
            data=0;
            nextnode=NULL;
            prevnode=NULL;
        }
        node(int value) {
            data=value;
            nextnode=NULL;
            prevnode=NULL;
        }
        node(int value, node* nn, node* pn) {
            data=value;
            nextnode=nn;
            prevnode=pn;
        }
};

class DLL {
        int nodecount=0;
        node* head=NULL;
    public:
        void insertAttail(int value) {
            if(head==NULL) {  // if array was empty
                node* n=new node(value);
                head=n;
                nodecount++;
                return;
            }
            node* temp=head;
            while(temp->nextnode!=NULL) {
                temp=temp->nextnode;
            }
            node* n=new node(value,NULL,temp);
            temp->nextnode=n;
            nodecount++;
        }
        void insertAthead(int value) {
            node* n=new node(value,head,NULL);
            if(head!=NULL){
                head->prevnode=n;
            }
```

```
54              head=n;
55              nodecount++;
56          }
57          void insertAtPos(int pos,int value) {
58              if(pos>nodecount-1){
59                  cout<<"Position more than nodes in list, Inserting at tail.\n";
60                  insertAttail(value);
61                  return;
62              }
63              int count=0;
64              node* temp=head;
65              while(temp->nextnode!=NULL && count<pos-1) {
66                  temp=temp->nextnode;
67                  count++;
68              }
69              node* n=new node(value,temp->nextnode,temp);
70              temp->nextnode=n;
71              n->nextnode->prevnode=n;
72              nodecount++;
73          }
74          void display() {
75              node* temp=head;
76              cout<<"HEAD | ";
77              while(temp!=NULL) {
78                  cout<<" <--"<<temp->prevnode<<"  |  "<<temp->data<<"  |  "<<temp->nextnode<<"
    --> ";
79                  temp=temp->nextnode;
80              }
81              cout<<"| TAIL"<<endl;
82          }
83  //Assuming ANY node means any of the 4 types (head,tail,position,value)
84          void deleteAtHead() {
85              if(head==NULL) {
86                  cout<<"Empty Linked List, Returning"<<endl;
87                  return;
88              }
89              node* todelete=head;
90              head=head->nextnode;
91              head->prevnode=NULL;
92              delete todelete;
93              nodecount--;
94          }
95          void deletion(int value) {
96              if(head==NULL) {
97                  cout<<"Empty Linked List, Returning"<<endl;
98                  return;
99              }
100             node* temp=head;
101             if(head->data==value) {
102                 deleteAtHead();
103                 return;
104             }
105             while(temp->data!=value) {
106                 if(temp->nextnode==NULL) {
107                     cout<<"Value not found, Returning\n";
108                     return;
```

```
109                 }
110                 temp=temp->nextnode;
111             }
112             if(temp->nextnode==NULL){
113                 deleteAtTail();
114                 return;
115             }
116             node* todelete=temp;
117             temp->prevnode->nextnode=temp->nextnode;
118             temp->nextnode->prevnode=temp->prevnode;
119             delete todelete;
120             nodecount--;
121         }
122         void deleteAtPos(int pos) {
123             if(pos==0){
124                 deleteAtHead();
125                 return;
126             }
127             else if(pos==nodecount-1){
128                 deleteAtTail();
129                 return;
130             }
131             if(pos>nodecount-1){
132                 cout<<"Invalid Position, Returning"<<endl;
133                 return;
134             }
135             if(head==NULL) {
136                 cout<<"Empty Linked List, Returning"<<endl;
137                 return;
138             }
139             int count=0;
140             node* temp=head;
141             while(temp->nextnode!=NULL && count<pos-1) {
142                 temp=temp->nextnode;
143                 count++;
144             }
145             node* todelete=temp->nextnode;
146             temp->nextnode=temp->nextnode->nextnode;
147             temp->nextnode->prevnode=temp;
148             delete todelete;
149             nodecount--;
150         }
151         void deleteAtTail() {
152             if(head==NULL) {
153                 cout<<"Empty Linked List, Returning"<<endl;
154                 return;
155             }
156             node* temp=head;
157             while(temp->nextnode!=NULL) {
158                 temp=temp->nextnode;
159             }
160             node* todelete=temp;
161             temp=temp->prevnode;
162             temp->nextnode=NULL;
163             delete todelete;
164             nodecount--;
```

```
165         }
166         void concatlist(DLL &obj){
167             node* temp=obj.head;
168             while(temp!=NULL){
169                 insertAttail(temp->data);
170                 temp=temp->nextnode;
171             }
172         }
173         void sortlist(){        //sorting by data , bubble sort
174             node* temp=head;
175             while(temp->nextnode!=NULL) {
176                 node* temp2=head;
177                 while(temp2->nextnode!=NULL) {
178                     if(temp2->data<temp2->nextnode->data) {
179                         char tempchar = temp2->data;
180                         temp2->data = temp2->nextnode->data;
181                         temp2->nextnode->data = tempchar;
182                     }
183                     temp2=temp2->nextnode;
184                 }
185                 temp=temp->nextnode;
186             }
187         }
188 };
189 
190 int main(){
191     DLL l,m;
192     for(int i=2;i<11;i+=2){  //Initializing L and M
193         l.insertAttail(i);
194         m.insertAttail(i-1);
195     }
196     cout<<"-----------LIST L (Evens)-------------------"<<endl;
197     l.display();
198     cout<<"-----------LIST M (odds)-------------------------"<<endl;
199     m.display();
200     cout<<"-------------------------------------"<<endl;
201     DLL n;
202     n.concatlist(l);
203     n.concatlist(m);
204     cout<<"-----------LIST N after concatenating L & M-------------------------"<<endl;
205     n.display();
206 
207     //Question 3 part starts here
208 
209     cout<<"-----------LIST N after Desc Sort-------------------------"<<endl;
210     n.sortlist();       //calls sorting function
211     n.display();
212 
213 }
```