

Question3.cpp

```
1 // <---Lab 04 - Doubly and Circular Linked List-->
2 /*3. Using the above created list N, sort the contents of list N in descending order.*/
3
4 #include<iostream>
5 using namespace std;
6
7 class node {
8     public:
9         int data;
10        node* nextnode; //to point to node after it
11        node* prevnode; //to point to node before it
12        node() {
13            data=0;
14            nextnode=NULL;
15            prevnode=NULL;
16        }
17        node(int value) {
18            data=value;
19            nextnode=NULL;
20            prevnode=NULL;
21        }
22        node(int value, node* nn, node* pn) {
23            data=value;
24            nextnode=nn;
25            prevnode=pn;
26        }
27 };
28
29 class DLL {
30     int nodecount=0;
31     node* head=NULL;
32     public:
33     void insertAttail(int value) {
34         if(head==NULL) { // if array was empty
35             node* n=new node(value);
36             head=n;
37             nodecount++;
38             return;
39         }
40         node* temp=head;
41         while(temp->nextnode!=NULL) {
42             temp=temp->nextnode;
43         }
44         node* n=new node(value,NULL,temp);
45         temp->nextnode=n;
46         nodecount++;
47     }
48     void insertAthead(int value) {
49         node* n=new node(value,head,NULL);
50         if(head!=NULL){
51             head->prevnode=n;
52         }
53         head=n;
```

```

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

```

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

```

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

```