

Question1.cpp

```
1  // <---Lab 04 - Doubly and Circular Linked List--->
2  // 1. Create a doubly link list and perform the mentioned tasks.
3  // a. Insert a new node at the end of the list.
4  // b. Insert a new node at the beginning of list.
5  // c. Insert a new node at given position.
6  // d. Delete any node.
7  // e. Print the complete doubly link list.
8
9
10 #include<iostream>
11 using namespace std;
12
13 class node {
14     public:
15         int data;
16         node* nextnode; //to point to node after it
17         node* prevnode; //to point to node before it
18         node() {
19             data=0;
20             nextnode=NULL;
21             prevnode=NULL;
22         }
23         node(int value) {
24             data=value;
25             nextnode=NULL;
26             prevnode=NULL;
27         }
28         node(int value, node* nn, node* pn) {
29             data=value;
30             nextnode=nn;
31             prevnode=pn;
32         }
33 };
34
35 class DLL {
36     int nodecount=0;
37     node* head=NULL;
38     public:
39     void insertAttail(int value) {
40         if(head==NULL) { // if list was empty
41             node* n=new node(value);
42             head=n;
43             nodecount++;
44             return;
45         }
46         node* temp=head;
47         while(temp->nextnode!=NULL) {
48             temp=temp->nextnode;
49         }
50         node* n=new node(value,NULL,temp);
51         temp->nextnode=n;
52         nodecount++;
53     }
```

```

54     void insertAtHead(int value) {
55         node* n=new node(value,head,NULL);
56         if(head!=NULL){
57             head->prevnode=n;
58         }
59         head=n;
60         nodecount++;
61     }
62     void insertAtPos(int pos,int value) {
63         if(pos<0){
64             cout<<"Position less than 0, Inserting at head.\n";
65             insertAtHead(value);
66             return;
67         }
68         if(pos>nodecount-1){
69             cout<<"Position more than nodes in list, Inserting at tail.\n";
70             insertAtTail(value);
71             return;
72         }
73         int count=0;
74         node* temp=head;
75         while(temp->nextnode!=NULL && count<pos-1) {
76             temp=temp->nextnode;
77             count++;
78         }
79         node* n=new node(value,temp->nextnode,temp);
80         temp->nextnode=n;
81         n->nextnode->prevnode=n;
82         nodecount++;
83     }
84     void display() {
85         node* temp=head;
86         cout<<"HEAD | ";
87         while(temp!=NULL) {
88             cout<<" <--"<<temp->prevnode<<" | "<<temp->data<<" | "<<temp->nextnode<<"--> ";
89             temp=temp->nextnode;
90         }
91         cout<<" | TAIL"<<endl;
92     }
93     //Assuming ANY node means any of the 4 types (head,tail,position,value)
94     void deleteAtHead() {
95         if(head==NULL) {
96             cout<<"Empty Linked List, Returning"<<endl;
97             return;
98         }
99         node* todelete=head;
100        head=head->nextnode;
101        head->prevnode=NULL;
102        delete todelete;
103        nodecount--;
104    }
105    void deletion(int value) {
106        if(head==NULL) {
107            cout<<"Empty Linked List, Returning"<<endl;
108            return;

```

```
109     }
110     node* temp=head;
111     if(head->data==value) {
112         deleteAtHead();
113         return;
114     }
115     while(temp->data!=value) {
116         if(temp->nextnode==NULL) {
117             cout<<"Value not found, Returning\n";
118             return;
119         }
120         temp=temp->nextnode;
121     }
122     if(temp->nextnode==NULL){
123         deleteAtTail();
124         return;
125     }
126     node* todelete=temp;
127     temp->prevnode->nextnode=temp->nextnode;
128     temp->nextnode->prevnode=temp->prevnode;
129     delete todelete;
130     nodecount--;
131 }
132 void deleteAtPos(int pos) {
133     if(pos<0){
134         cout<<"Position less than zero, INVALID. Returning..."<<endl;
135         return;
136     }
137     if(pos==0){
138         deleteAtHead();
139         return;
140     }
141     else if(pos==nodecount-1){
142         deleteAtTail();
143         return;
144     }
145     if(pos>nodecount-1){
146         cout<<"Invalid Position, Returning"<<endl;
147         return;
148     }
149     if(head==NULL) {
150         cout<<"Empty Linked List, Returning"<<endl;
151         return;
152     }
153     int count=0;
154     node* temp=head;
155     while(temp->nextnode!=NULL && count<pos-1) {
156         temp=temp->nextnode;
157         count++;
158     }
159     node* todelete=temp->nextnode;
160     temp->nextnode=temp->nextnode->nextnode;
161     temp->nextnode->prevnode=temp;
162     delete todelete;
163     nodecount--;
164 }
```

```

165     void deleteAtTail() {
166         if(head==NULL) {
167             cout<<"Empty Linked List, Returning"<<endl;
168             return;
169         }
170         node* temp=head;
171         while(temp->nextnode!=NULL) {
172             temp=temp->nextnode;
173         }
174         node* todelete=temp;
175         temp=temp->prevnode;
176         temp->nextnode=NULL;
177         delete todelete;
178         nodecount--;
179     }
180 };
181
182 int main() {
183     DLL list;
184     float input=0;
185     int value;
186     while(input!=0.5) {
187         cout<<"-----\n";
188         cout<<"CURRENT LINKED LIST:\n";
189         list.display();
190         cout<<"-----\n";
191         cout<<"What would you like to do with the linked list?\n";
192         cout<<"1. Insert\t2. Delete\nEnter 0.5 to Exit\n[Anything else will default to Delete]\n";
193         cin>>input;
194         if(input==1) {
195             cout<<"Enter Value to insert: ";
196             cin>>value;
197             cout<<"Where to Insert in Linked List?\n";
198             cout<<"1. At head\t2. At tail\t3. At specified Position\n[Any other value will default
to Insertion at Head]\n";
199             cin>>input;
200             if(input == 2){
201                 list.insertAttail(value);
202             }
203             else if(input == 3){
204                 int pos;
205                 cout<<"Enter the Position to insert into: ";
206                 cin>>pos;
207                 list.insertAtPos(pos,value);
208             }
209             else{
210                 list.insertAthead(value);
211             }
212         }
213         else if(input==0.5){
214             break;
215         }
216         else{
217             cout<<"Where to Delete from Linked List?\n";
218             cout<<"1. At head\t2. At tail\t3. At specified Position\t4. Delete a specific
Value\n[Any other value will default to Deletion from Head]\n";

```

```
220         cin>>input;
221         if(input == 2){
222             list.deleteAtTail();
223         }
224         else if(input == 3){
225             int pos;
226             cout<<"Enter the Position to Delete from: ";
227             cin>>pos;
228             list.deleteAtPos(pos);
229         }
230         else if(input == 4){
231             int pos;
232             cout<<"Enter the Value to Delete: ";
233             cin>>value;
234             list.deletion(value);
235         }
236         else{
237             list.deleteAtHead();
238         }
239     }
240 }
241 }
242 }
```