

Question4.cpp

```
1 // <---Lab 04 - Doubly and Circular Linked List--->
2 /*4. Create a circular 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* next;
17         node(int value) {
18             data=value;
19             next=NULL;
20         }
21         node(int value,node* nxt) {
22             data=value;
23             next=nxt;
24         }
25 };
26
27 class CLL{
28     node* head=NULL;
29     int nodecount=0;
30     public:
31     void appendNode(int value){ //insert at end of list / tail
32         if(head==NULL) { //Check if Linked List Empty.
33             node* n=new node(value,n);
34             head=n;
35             head->next=head;
36             nodecount++;
37             return;
38         }
39         else{
40             node* temp=head;
41             while(temp->next!=head) {
42                 temp=temp->next;
43             }
44             node* n=new node(value,head);
45             temp->next=n;
46             nodecount++;
47         }
48     }
49     void prependNode(int value){ //insert at start of list / head
50         if(head==NULL) { //Check if Linked List Empty.
51             node* n=new node(value,n);
52             head=n;
53             head->next=head;
```

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

```

110         prev=temp;
111         temp=temp->next;
112     }
113     if(temp->data==value){
114         prev->next=temp->next; //skip temp (i.e delete)
115         delete temp;
116     }
117     if(temp->next==head){
118         cout<<"Value not in Linked List.\n";
119         return;
120     }
121 }
122 void updateNodeByKey(int value){ // update by value
123     node* temp=head;
124     while(temp->next!=head&&temp->data!=value) {
125         node* prev=temp;
126         temp=temp->next;
127     }
128     if(temp->data==value){
129         cout<<"Enter new value: ";
130         cin>>temp->data;
131     }
132     if(temp->next==head){
133         cout<<"Value not in Linked List.\n";
134         return;
135     }
136 }
137 void print(){
138     node* temp=head;
139     cout<<"[HEAD] ";
140     if(head!=NULL){
141         cout<<temp->data<<" | "<<temp->next<<" -> ";
142         temp=temp->next;
143         while(temp!=head) {
144             cout<<temp->data<<" | "<<temp->next<<" -> ";
145             temp=temp->next;
146         }
147         cout<<"head [TAIL]"<<endl;
148     }
149     else{
150         cout<<"NULL [TAIL]"<<endl;
151     }
152 }
153 };
154
155 int main(){
156     CLL list;
157     float input=0;
158     int value;
159     while(input!=0.5) {
160         cout<<"-----\n";
161         cout<<"CURRENT LINKED LIST:\n";
162         list.print();
163         cout<<"-----\n";
164         cout<<"What would you like to do with the linked list?\n";
165         cout<<"1. Insert\t2. Delete\t3.Update\nEnter 0.5 to Exit\n[Anything else will default to

```

```

Delete]\n";
166     cin>>input;
167     if(input==1) {
168         cout<<"Enter Value to insert: ";
169         cin>>value;
170         cout<<"Where to Insert in Linked List?\n";
171         cout<<"1. At head\t2. At tail\t3. At specified Position\n[Any other value will default
to Insertion at Head]\n";
172         cin>>input;
173         if(input == 2){
174             list.appendNode(value);
175         }
176         else if(input == 3){
177             int pos;
178             cout<<"Enter the Position to insert After: ";
179             cin>>pos;
180             list.insertNodeAfter(pos,value);
181         }
182         else{
183             list.prependNode(value);
184         }
185     }
186 }
187 else if(input==0.5){
188     break;
189 }
190 else if(input==3){
191     cout<<"Enter the Value to Update: ";
192     cin>>value;
193     list.updateNodeByKey(value);
194 }
195 else{
196     cout<<"Where to Delete from Linked List?\n";
197     cout<<"1. At head\t2. At tail\t3. Delete a specific Value\n[Any other value will
default to Deletion from Head]\n";
198     cin>>input;
199     if(input == 2){
200         list.deleteattail();
201     }
202     else if(input == 3){
203         int pos;
204         cout<<"Enter the Value to Delete: ";
205         cin>>value;
206         list.deleteNodeByKey(value);
207     }
208     else{
209         list.deleteathead();
210     }
211 }
212 }
213 }
214 }

```