1/12/24, 9:31 PM Question4.cpp

Question4.cpp

```
1 // <----Lab 04 - Doubly and Circular Linked List---->
 3
   // Q4. Create a circular link list and perform the mentioned tasks.
   // a. Insert a new node at the end of the list.
   // b. Insert a new node at the beginning of list.
   // c. Insert a new node at given position.
6
 7
   // d. Delete any node.
   // e. Print the complete doubly link list.
8
9
   #include<iostream>
10
11
    using namespace std;
12
13
    class node {
14
        public:
15
            int data;
            node* next;
16
            node(int value) {
17
18
                data=value;
                next=NULL;
19
20
            }
            node(int value,node* nxt) {
21
22
                data=value;
23
                next=nxt;
24
            }
25
   };
26
27
    class CLL{
28
        node* head=NULL;
29
        int nodecount=0;
30
        public:
        void appendNode(int value){ //insert at end of list / tail
31
            if(head==NULL) { //Check if Linked List Empty.
32
33
                node* n=new node(value,n);
34
                head=n;
35
                head->next=head;
36
                nodecount++;
37
                return;
38
            }
            else{
39
                node* temp=head;
40
41
                while(temp->next!=head) {
42
                    temp=temp->next;
43
                node* n=new node(value,head);
44
45
                temp->next=n;
                nodecount++;
46
47
48
        void prependNode(int value){ //insert at start of list / head
49
50
            if(head==NULL) { //Check if Linked List Empty.
51
                node* n=new node(value,n);
52
                head=n;
                head->next=head;
53
```

```
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
              temp->next=n;
 62
 63
              head=n:
 64
              nodecount++;
 65
         void insertNodeAfter(int posvalue,int value){  // insert at position (i+1)
 66
 67
              if(head==NULL) { //Check if Linked List Empty.
                  cout<<"Empty List, adding at Head.\n";</pre>
 68
 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";</pre>
 76
                  appendNode(value);
 77
                  return;
 78
              }
 79
              int count=0;
              node* temp=head;
 80
 81
             while(temp->next!=head&&count<posvalue) {</pre>
 82
                  temp=temp->next;
 83
                  count++;
 84
              }
              node* n=new node(value,temp->next);
 85
 86
             temp->next=n;
 87
         }
 88
         void deleteathead(){
 89
              node* temp=head;
 90
             while(temp->next=head){
                  temp=temp->next;
 91
 92
 93
              temp->next=head->next;
 94
              delete head;
 95
             head = temp->next;
96
 97
         void deleteattail(){
             node* temp=head;
 98
 99
             while(temp->next->next!=head){
100
                  temp=temp->next;
101
102
              node* todelete=temp->next;
103
              temp->next=head;
104
              delete todelete;
105
         void deleteNodeByKey(int value){ // delete by value
106
             node* temp=head;
107
108
              node* prev=temp;
             while(temp->next!=head&&temp->data!=value) {
109
```

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

```
to Delete]\n";
166
              cin>>input;
167
              if(input==1) {
                  cout<<"Enter Value to insert: ";</pre>
168
169
                  cin>>value:
170
                  cout<<"Where to Insert in Linked List?\n";</pre>
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){
                      list.appendNode(value);
174
175
176
                  else if(input == 3){
177
                      int pos;
178
                      cout<<"Enter the Position to insert After: ";</pre>
179
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){
                  cout<<"Enter the Value to Update: ";</pre>
191
192
                  cin>>value;
193
                  list.updateNodeByKey(value);
194
              }
              else{
195
196
                  cout<<"Where to Delete from Linked List?\n";</pre>
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
                  else if(input == 3){
202
                      int pos;
203
204
                      cout<<"Enter the Value to Delete: ";</pre>
205
                      cin>>value;
206
                      list.deleteNodeByKey(value);
207
                  }
208
                  else{
209
                      list.deleteathead();
210
211
              }
212
213
         }
214
    }
```