Question4.cpp

```cpp
// <---Lab 04 - Doubly and Circular Linked List--->
/*4. 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.
d. Delete any node.
e. Print the complete doubly link list.*/


#include<iostream>
using namespace std;

class node {
    public:
        int data;
        node* next;
        node(int value) {
            data=value;
            next=NULL;
        }
        node(int value,node* nxt) {
            data=value;
            next=nxt;
        }
};

class CLL{
    node* head=NULL;
    int nodecount=0;
    public:
    void appendNode(int value){ //insert at end of list / tail
        if(head==NULL) {  //Check if Linked List Empty.
            node* n=new node(value,n);
            head=n;
            head->next=head;
            nodecount++;
            return;
        }
        else{
            node* temp=head;
            while(temp->next!=head) {
                temp=temp->next;
            }
            node* n=new node(value,head);
            temp->next=n;
            nodecount++;
        }
    }
    void prependNode(int value){ //insert at start of list / head
        if(head==NULL) {  //Check if Linked List Empty.
            node* n=new node(value,n);
            head=n;
            head->next=head;
```

```
            nodecount++;
            return;
        }
        node* n=new node(value,head);
        node* temp=head;
        while(temp->next!=head) {
            temp=temp->next;
        }
        temp->next=n;
        head=n;
        nodecount++;
    }
    void insertNodeAfter(int posvalue,int value){  // insert at position (i+1)
        if(head==NULL) {  //Check if Linked List Empty.
            cout<<"Empty List, adding at Head.\n";
            node* n=new node(value,n);
            head=n;
            nodecount++;
            return;
        }
        if(posvalue>nodecount-1){
            cout<<"Position more than nodes in list, Inserting at tail.\n";
            appendNode(value);
            return;
        }
        int count=0;
        node* temp=head;
        while(temp->next!=head&&count<posvalue) {
            temp=temp->next;
            count++;
        }
        node* n=new node(value,temp->next);
        temp->next=n;
    }
    void deleteathead(){
        node* temp=head;
        while(temp->next=head){
            temp=temp->next;
        }
        temp->next=head->next;
        delete head;
        head = temp->next;
    }
    void deleteattail(){
        node* temp=head;
        while(temp->next->next!=head){
            temp=temp->next;
        }
        node* todelete=temp->next;
        temp->next=head;
        delete todelete;
    }
    void deleteNodeByKey(int value){  // delete by value
        node* temp=head;
        node* prev=temp;
        while(temp->next!=head&&temp->data!=value) {
```

```
            prev=temp;
            temp=temp->next;
        }
        if(temp->data==value){
            prev->next=temp->next;  //skip temp (i.e delete)
            delete temp;
        }
        if(temp->next==head){
            cout<<"Value not in Linked List.\n";
            return;
        }
    }
    void updateNodeByKey(int value){  // update by value
        node* temp=head;
        while(temp->next!=head&&temp->data!=value) {
            node* prev=temp;
            temp=temp->next;
        }
        if(temp->data==value){
            cout<<"Enter new value: ";
            cin>>temp->data;
        }
        if(temp->next==head){
            cout<<"Value not in Linked List.\n";
            return;
        }
    }
    void print(){
        node* temp=head;
        cout<<"[HEAD] ";
        if(head!=NULL){
        cout<<temp->data<<"| "<<temp->next<<" -> ";
        temp=temp->next;
        while(temp!=head) {
            cout<<temp->data<<"| "<<temp->next<<" -> ";
            temp=temp->next;
        }
        cout<<"head [TAIL]"<<endl;
        }
        else{
            cout<<"NULL [TAIL]"<<endl;
        }
    }
};

int main(){
    CLL list;
    float input=0;
    int value;
    while(input!=0.5) {
        cout<<"-----------------------------------------------------\n";
        cout<<"CURRENT LINKED LIST:\n";
        list.print();
        cout<<"-----------------------------------------------------\n";
        cout<<"What would you like to do with the linked list?\n";
        cout<<"1. Insert\t2. Delete\t3.Update\nEnter 0.5 to Exit\n[Anything else will default to
```

```
Delete]\n";
        cin>>input;
        if(input==1) {
            cout<<"Enter Value to insert: ";
            cin>>value;
            cout<<"Where to Insert in Linked List?\n";
            cout<<"1. At head\t2. At tail\t3. At specified Position\n[Any other value will default to Insertion at Head]\n";
            cin>>input;
            if(input == 2){
                list.appendNode(value);
            }
            else if(input == 3){
                int pos;
                cout<<"Enter the Position to insert After: ";
                cin>>pos;
                list.insertNodeAfter(pos,value);
            }
            else{
                list.prependNode(value);
            }

        }
        else if(input==0.5){
            break;
        }
        else if(input==3){
            cout<<"Enter the Value to Update: ";
            cin>>value;
            list.updateNodeByKey(value);
        }
        else{
            cout<<"Where to Delete from Linked List?\n";
            cout<<"1. At head\t2. At tail\t3. Delete a specific Value\n[Any other value will default to Deletion from Head]\n";
            cin>>input;
            if(input == 2){
                list.deleteattail();
            }
            else if(input == 3){
                int pos;
                cout<<"Enter the Value to Delete: ";
                cin>>value;
                list.deleteNodeByKey(value);
            }
            else{
                list.deleteathead();
            }
        }

    }
}
```