DSA Assignment 6

Name: Jashnoor Singh
Roll Number: 1024030106
Batch: 2C35

Circular Linked list template

```
#include <iostream>
using namespace std;
class node_cll{
    public:
    int data;
    node_cll* next;
    node_cll(int d){
        data = d;
        next = this; // makes next point to current obj
    }
};

void insert(node_cll* head, int data){
    node_cll* temp = new node_cll(data);
    temp->next = head->next;
    head->next = temp;
}
void display(node_cll* head){
    if(head->next == head){
        cout<<head->data;
        return;
    }
    node_cll* temp = head;
    cout<<temp->data<<' ';
    temp = temp->next;
    while(temp!=head){
        cout<<temp->data<<' ';
        temp = temp->next;
    }
}
```

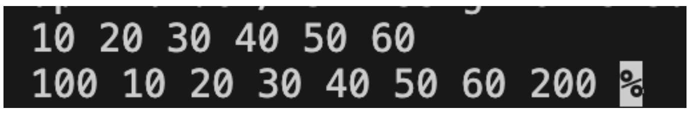
Doubly Linked List Template

```
// here we will make a double linked list class for code reusability

#include <iostream>
using namespace std;

class node{
    public:
    int data;
    node* left;
    node* right;
    node(int d){
        data = d;
        left = right = NULL;
    }
};
void display(node* head){
    while(head!=NULL){
        cout<<head->data<<' ';
        head = head->right;
    }
}
```

Doubly Linked list Q1
(A)

```
// insertion cases

#include "dll.cpp"

void insertTail(node* &tail, int data){
    node* temp = new node(data);
    temp->left = tail;
    tail->right = temp;
    tail = temp;
}
void insertHead(node* &head, int data){
    node* temp = new node(data);
    temp->right = head;
    head->left = temp;
    head = temp;
}
void insertPosition(node* &head, int data, int pos){
    int count = 1;
    node* temp_head = head;
    if(pos == 1){
        insertHead(head,data);
        return;
    }
    while(count!=(pos-1)){
        temp_head = temp_head->right;
        count++;
    }
    if(temp_head->right == NULL){ // if last position insertion
        insertTail(temp_head,data);
        return;
    }
    node* temp = new node(data);
    temp->left = temp_head;
    temp->right = temp_head->right;
    temp_head->right->left = temp;
    temp_head->right = temp;
}

int main(){
    node* head = new node(10); // making first node with value 10
    node* tail = head;
    insertTail(tail,20);
    insertTail(tail,30);
    insertTail(tail,40);
    insertTail(tail,50);
    insertTail(tail,60);
    display(head);
    cout<<endl;
    insertPosition(head,100,1);
    insertPosition(head,200,8);
    display(head);
}
```

Output:

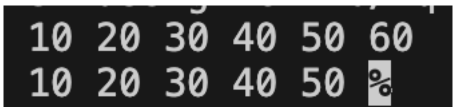


```
10 20 30 40 50 60
100 10 20 30 40 50 60 200 %
```

(B)

```
// deletion cases for doubly linked list

#include "dll.cpp"

void insertTail(node* &tail, int data){
    node* temp = new node(data);
    temp->left = tail;
    tail->right = temp;
    tail = temp;
}
void del_first(node* &head){ // deletion of first node
    if(head->right == NULL){ // if only one node exists
        delete head;
        head = NULL;
        return;
    }
    node* temp = head;
    head = temp->right;
    head->left = NULL;
    delete temp;
}
void del(node* &head, node* &tail, int index){
    if(head == NULL){
        cout<<"list is already empty";
        return;
    }
    if(index==1){
        del_first(head);
        return;
    }
    int count = 1;
    node* temp = head;
    while(count!=index-1){
        temp = temp->right;
        count++;
    }
    node* temp2 = temp->right;
    if(temp2->right == NULL){
        temp2->left = NULL;
        temp->right = NULL;
        tail = temp;
        delete temp2;
        return;
    }
    temp->right = temp2->right;
    temp->right->left = temp2->left;
    temp2->right = NULL;
    temp2->left = NULL;
    delete temp2;
}
int main(){
    node* head = new node(10); // making first node with value 10
    node* tail = head;
    insertTail(tail,20);
    insertTail(tail,30);
    insertTail(tail,40);
    insertTail(tail,50);
    insertTail(tail,60);
    display(head);
    cout<<endl;
    del(head,tail,6);
    display(head);
}
```

Output:



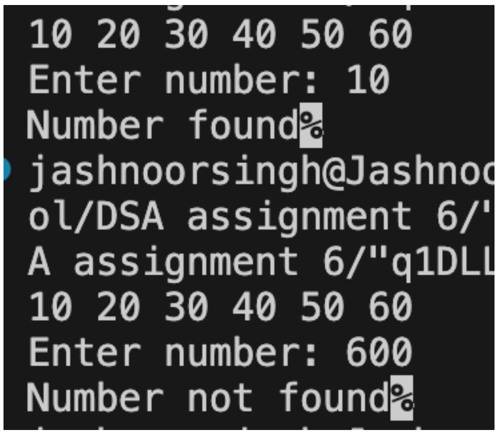
```
10 20 30 40 50 60
10 20 30 40 50 %
```

(C)

// searching in DLL

```
#include "dll.cpp"
void insertTail(node* &tail, int data){
    node* temp = new node(data);
    temp->left = tail;
    tail->right = temp;
    tail = temp;
}
bool search(node* head, int key){
    while(head!=NULL){
        if(head->data == key)return true;
        head=head->right;
    }
    return false;
}
int main(){
    node* head = new node(10); // making first node with value 10
    node* tail = head;
    insertTail(tail,20);
    insertTail(tail,30);
    insertTail(tail,40);
    insertTail(tail,50);
    insertTail(tail,60);
    display(head);
    cout<<endl;
    if(search(head,60)){
        cout<<"Number found";
    }
    else cout<<"Number not found";
}
```

Output:



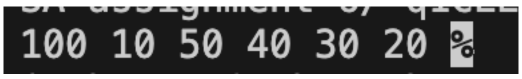
```
10 20 30 40 50 60
Enter number: 10
Number found%
jashnoorsingh@Jashnoo
ol/DSA assignment 6/'
A assignment 6/"q1DLL
10 20 30 40 50 60
Enter number: 600
Number not found%
```

Circular Linked list Q1

(A)

```
#include "cll.cpp"

void insert_anywh(node_cll* &head, int data, int pos, int size){
    if(pos == 1 || pos == size){
        node_cll* temp = head->next;
        while(temp->next != head){
            temp = temp->next;
        }
        node_cll* temp2 = new node_cll(data);
        temp->next = temp2;
        temp2->next = head;
        if(pos==1)head = temp2; // if we make this new head then we insert at front or else
we insert at back
    }
    else{
        int count = 1;
        node_cll* temp = head;
        while(count != pos-1){
            temp = temp->next;
            count++;
        }
        node_cll* temp2 = new node_cll(data);
        temp2->next = temp->next;
        temp->next = temp2;
    }
}

int main(){
    node_cll* head = new node_cll(10);
    insert(head,20);
    insert(head,30);
    insert(head,40);
    insert(head,50);
    node_cll* temp = head->next;
    int size = 1;
    while(temp!=head){ // calculating size of linked list
        size++;
        temp = temp->next;
    }
    insert_anywh(head,100,1,5);
    display(head);
}
```

Output:

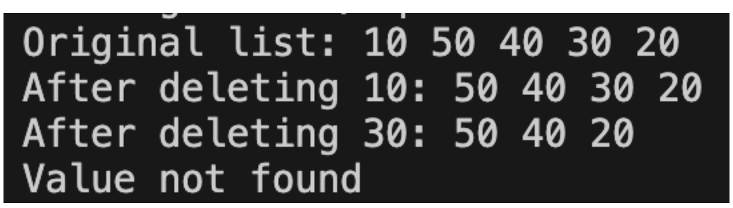


```
100 10 50 40 30 20 %
```

(B)

```
// delete a node by value

#include "cll.cpp"

void deleteNode(node_cll*& head, int value){
    if (head == nullptr){
        cout << "List is empty\n";
        return;
    }
    node_cll* curr = head;
    node_cll* prev = nullptr;
    if (head->next == head && head->data == value) {
        delete head;
        head = nullptr;
        return;
    }
    do{
        if (curr->data == value) break;
        prev = curr;
        curr = curr->next;
    }while(curr != head);
    if(curr->data != value){
        cout<<"Value not found\n";
        return;
    }
    if (curr == head){
        node_cll* last = head;
        while(last->next != head)
            last = last->next;
        last->next = head->next;
        head = head->next;
        delete curr;
    }
    else {
        prev->next = curr->next;
        delete curr;
    }
}

int main(){
    node_cll* head = new node_cll(10);
    insert(head,20);
    insert(head,30);
    insert(head,40);
    insert(head,50);
    cout << "Original list: ";
    display(head);
    cout<<endl;
    deleteNode(head,10); // deleting head
    cout << "After deleting 10: ";
    display(head);
    cout<<endl;
    deleteNode(head,30); // deleting middle node
    cout << "After deleting 30: ";
    display(head);
    cout<<endl;
    deleteNode(head, 99); // deleting non-existent value
}
```

Output:

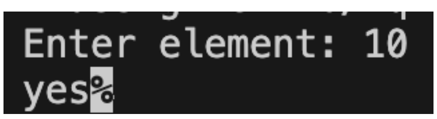


```
Original list: 10 50 40 30 20
After deleting 10: 50 40 30 20
After deleting 30: 50 40 20
Value not found
```

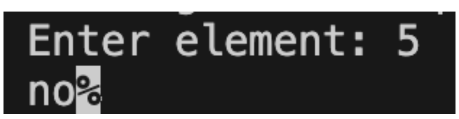
(C)

```
// searching in cll

#include "cll.cpp"

bool search(node_cll* head,int key){
    if(head->data == key)return true;
    node_cll* temp = head->next;
    while(temp!=head){
        if(temp->data == key)return true;
        temp = temp->next;
    }
    return false;
}

int main(){
    node_cll* head = new node_cll(10);
    insert(head,20);
    insert(head,30);
    insert(head,40);
    insert(head,50);
    insert(head,60);
    if(search(head,10))cout<<"yes";
    else cout<<"no";
}
```

Output:



```
Enter element: 10
yes%
```

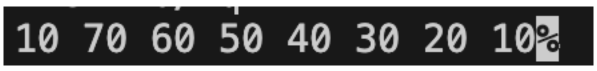


```
Enter element: 5
no%
```

Q2

```
#include <iostream>
using namespace std;
class node_cll{
    public:
    int data;
    node_cll* next;
    node_cll(int d){
        data = d;
        next = this; // makes next point to current obj
    }
};

void insert(node_cll* head, int data){
    node_cll* temp = new node_cll(data);
    temp->next = head->next;
    head->next = temp;
}
void display(node_cll* head){
    if(head->next == head){
        cout<<head->data;
        return;
    }
    node_cll* temp = head;
    while(1){
        cout<<temp->data<<' ';
        temp = temp->next;
        if(temp == head){
            cout<<temp->data;
            break;
        }
    }
}int main(){
    node_cll* head = new node_cll(10);
    insert(head,20);
    insert(head,30);
    insert(head,40);
    insert(head,50);
    insert(head,60);
    insert(head,70);
    display(head);
}
```

Output

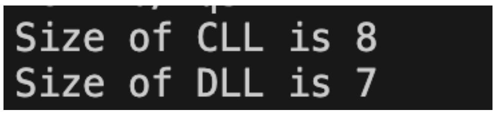


```
10 70 60 50 40 30 20 10%
```

Q3

```
#include "dll.cpp"
#include "cll.cpp"

void insertTail(node* &tail, int data){
    node* temp = new node(data);
    temp->left = tail;
    tail->right = temp;
    tail = temp;
}
int size_cll(node_cll* head){
    int n=1;
    node_cll* temp = head->next;
    while(temp!=head){
        n++;
        temp=temp->next;
    }
    return n;
}
int size_dll(node* head){
    int n=1;
    node* temp = head->right;
    while(temp!=NULL){
        n++;
        temp=temp->right;
    }
    return n;
}

int main(){
    node_cll* head = new node_cll(10); // making cll
    insert(head,20);
    insert(head,30);
    insert(head,40);
    insert(head,50);
    insert(head,60);
    insert(head,70);
    insert(head,80);
    node* head2 = new node(10);
    node* tail = head2;
    insertTail(tail,20);
    insertTail(tail,30);
    insertTail(tail,40);
    insertTail(tail,50);
    insertTail(tail,60);
    insertTail(tail,70);
    cout<<"Size of CLL is "<<size_cll(head)<<endl;
    cout<<"Size of DLL is "<<size_dll(head2)<<endl;
}
```

Output

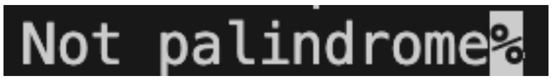


```
Size of CLL is 8
Size of DLL is 7
```

Q4

```cpp
#include <iostream>
using namespace std;
class node{
    public:
    char data;
    node* left;
    node* right;
    node(char d){
        data = d;
        left = right = NULL;
    }
};

void insertTail(node* &tail, char data){
    node* temp = new node(data);
    temp->left = tail;
    tail->right = temp;
    tail = temp;
}
bool isPalindrome(node* head, node* tail){
    while(head!=tail){
        if(head->data!=tail->data)return false;
        head = head->right;
        tail = tail->left;
    }
    return true;
}

int main(){
    node* head = new node('a'); // making first node with value 10
    node* tail = head;
    insertTail(tail,'b');
    insertTail(tail,'c');
    insertTail(tail,'b');
    insertTail(tail,'d');
    if(isPalindrome(head,tail)){
        cout<<"Is palindrome";
    }
    else cout<<"Not palindrome";
}
```

Output:

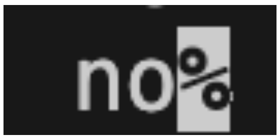


```
Not palindrome%
```

Q5

```
// Wap to check if list is circular linked list or not
#include "cll.cpp"

bool check_cll(node_cll* head){
    if(head->next == head)return true; // for single element list
    node_cll* temp = head->next;
    if(temp == NULL)return false;
    while(temp!=head){
        if(temp->next == NULL)return false;
        temp=temp->next;
    }
    return true;
}

int main(){
    node_cll* head = new node_cll(10); // making a node
    insert(head,20);
    insert(head,30);
    insert(head,40);
    insert(head,50);
    insert(head,60);
    insert(head,70);
    insert(head,80);
    head->next = NULL; // making this list singly linked list
    if(check_cll(head))cout<<"yes";
    else cout<<"no";
}
```

Output:



```
no%
```