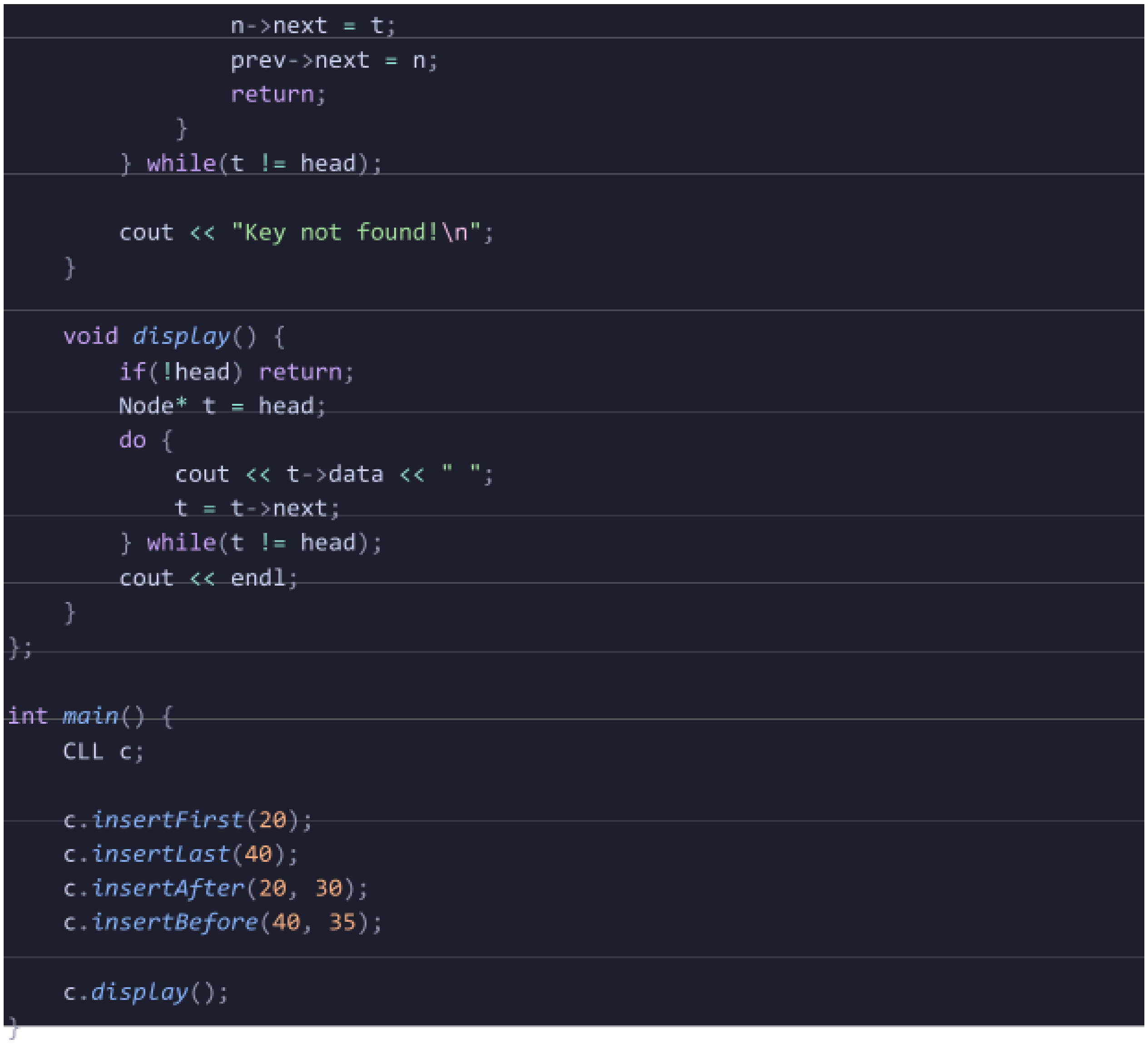
Data Structures (UCS301) Assignment

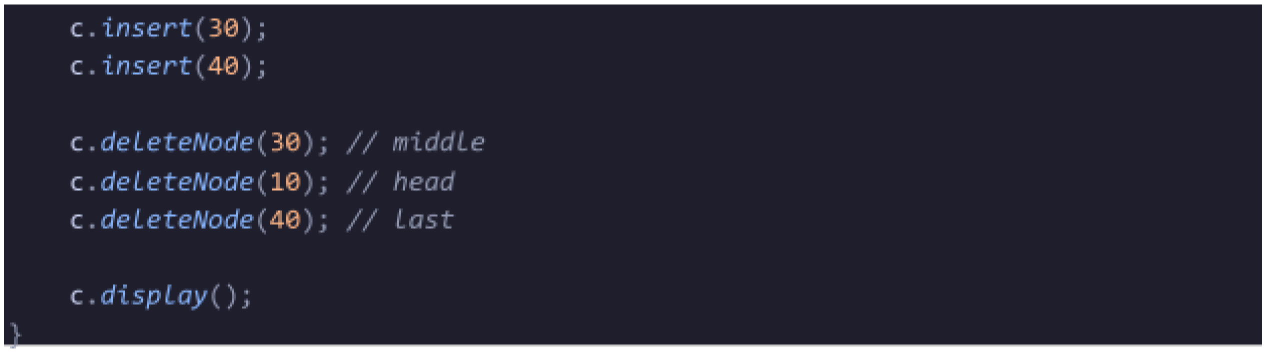
NAME-Suneet Arora **ROLL NO-1024030174**

```cpp
#include <bits/stdc++.h>
using namespace std;

class CLL {
public:
    struct Node {
        int data;
        Node* next;
        Node(int x) : data(x), next(NULL) {}
    };

    Node* head = NULL;

    // Insert at first
    void insertFirst(int x) {
        Node* n = new Node(x);

        if(!head) {
            head = n;
            n->next = head;
            return;
        }

        Node* t = head;
        while(t->next != head) t = t->next;

        n->next = head;
        t->next = n;
        head = n;
    }

    // Insert at last
    void insertLast(int x) {
        Node* n = new Node(x);

        if(!head) {
            head = n;
            n->next = head;
            return;
        }
        Node* t = head;
        while(t->next != head) t = t->next;

        t->next = n;
        n->next = head;
    }

    // Insert after a specific node
    void insertAfter(int key, int x) {
        if(!head) return;

        Node* t = head;
        do {
            if(t->data == key) { Node*
                n = new Node(x);
                n->next = t->next;
                t->next = n;
                return;
            }
            t = t->next;
        } while(t != head);

        cout << "Key not found!\n";
    }

    // Insert before a specific node
    void insertBefore(int key, int x) {
        if(!head) return;

        // If key is head
        if(head->data == key) {
            insertFirst(x);
            return;
        }

        Node* t = head;
        Node* prev = NULL;

        do {
            prev = t;
            t = t->next;

            if(t->data == key) { Node*
                n = new Node(x);
```

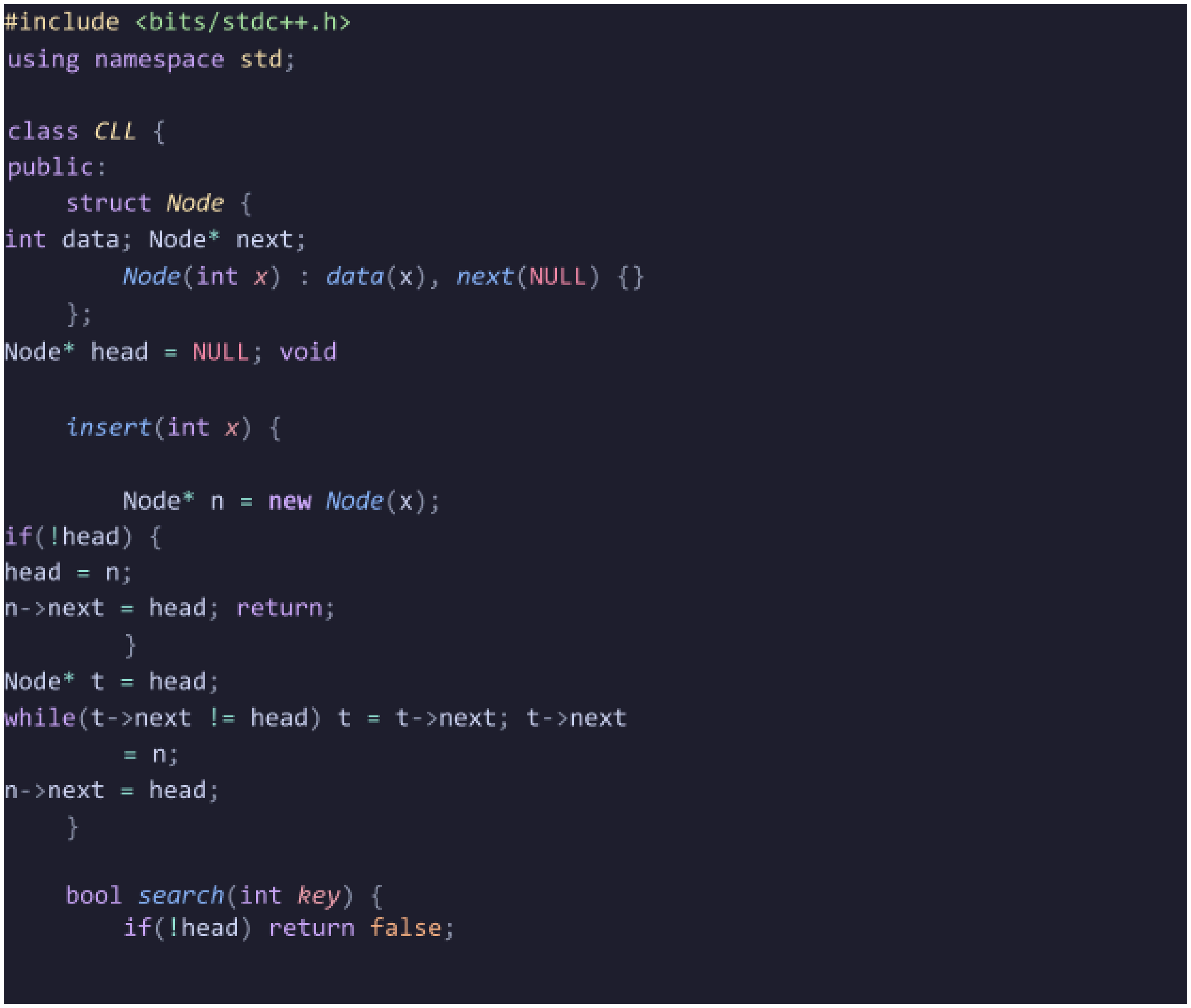


```
                n->next = t;
                prev->next = n;
                return;
            }
        } while(t != head);

        cout << "Key not found!\n";
    }

    void display() {
        if(!head) return;
        Node* t = head;
        do {
            cout << t->data << " ";
            t = t->next;
        } while(t != head);
        cout << endl;
    }
};

int main() {
    CLL c;

    c.insertFirst(20);
    c.insertLast(40);
    c.insertAfter(20, 30);
    c.insertBefore(40, 35);

    c.display();
}
```

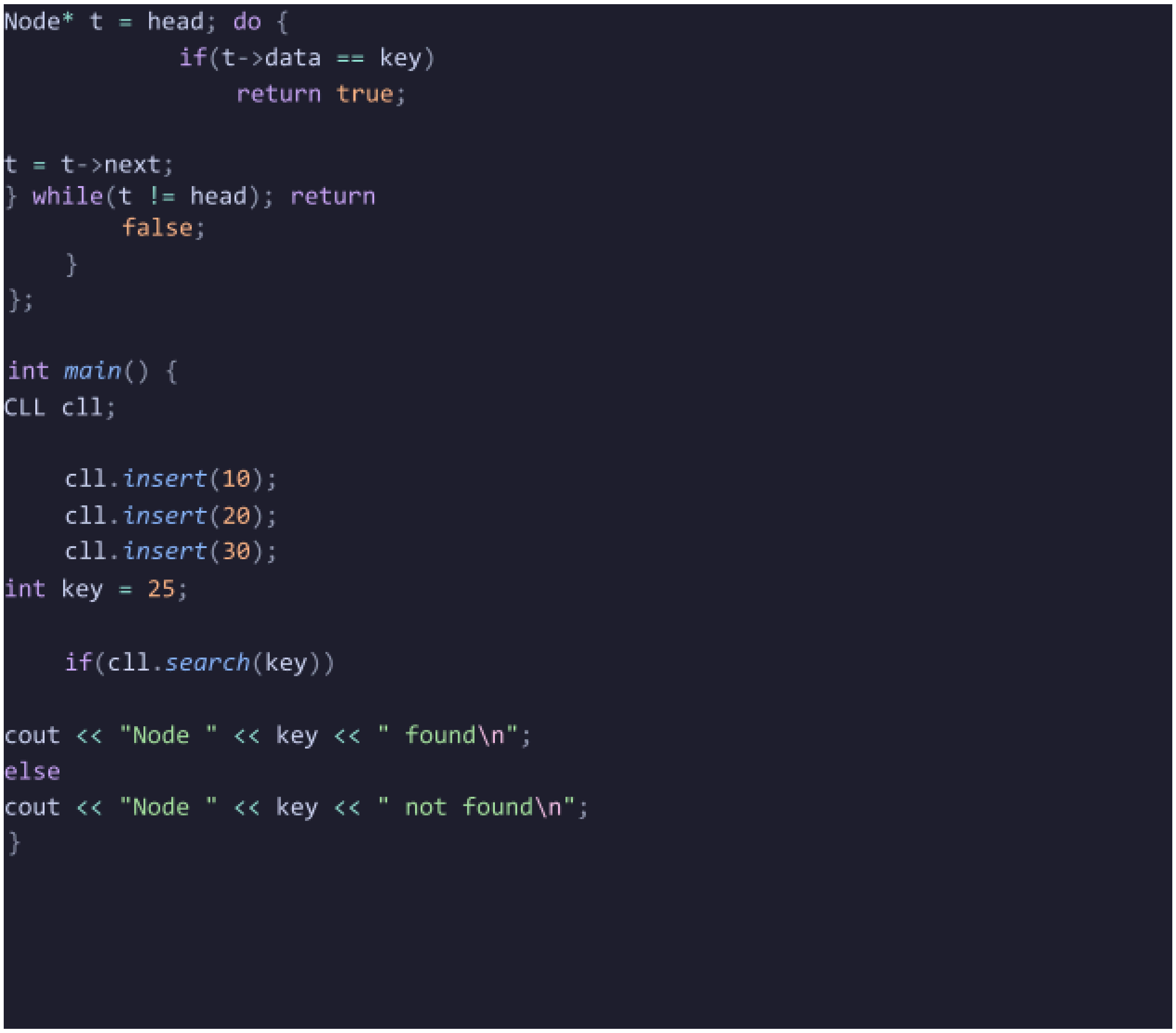
```cpp
#include <bits/stdc++.h>
using namespace std;

class CLL {
public:
    struct Node {
        int data;
        Node* next;
        Node(int x) : data(x), next(NULL) {}
    };

    Node* head = NULL;

    void insert(int x) {
        Node* n = new Node(x);
        if(!head) {
            head = n;
            n->next = head;
            return;
        }
        Node* t = head;
        while(t->next != head) t = t->next;
        t->next = n;
        n->next = head;
    }

    void deleteNode(int key) {
        if(!head) return;

        Node* curr = head;
        Node* prev = NULL;

        // Case 1: head node delete
        if(curr->data == key) {

            // Only one node
            if(curr->next == head) {
                head = NULL;
                delete curr;
                return;
            }

            // Move to last node
            while(curr->next != head) curr = curr->next;
            // curr is last node
            curr->next = head->next;
            Node* temp = head;
            head = head->next;

            delete temp;
            return;
        }

        // Case 2: delete middle/last
        prev = head;
        curr = head->next;

        while(curr != head && curr->data != key) {
            prev = curr;
            curr = curr->next;
        }

        if(curr == head) {
            cout << "Node not found!\n";
            return;
        }

        prev->next = curr->next;
        delete curr;
    }

    void display() {
        if(!head) return;

        Node* t = head;
        do {
            cout << t->data << " ";
            t = t->next;
        } while(t != head);

        cout << endl;
    }
};

int main() {
    CLL c;

    c.insert(10);
    c.insert(20);
```



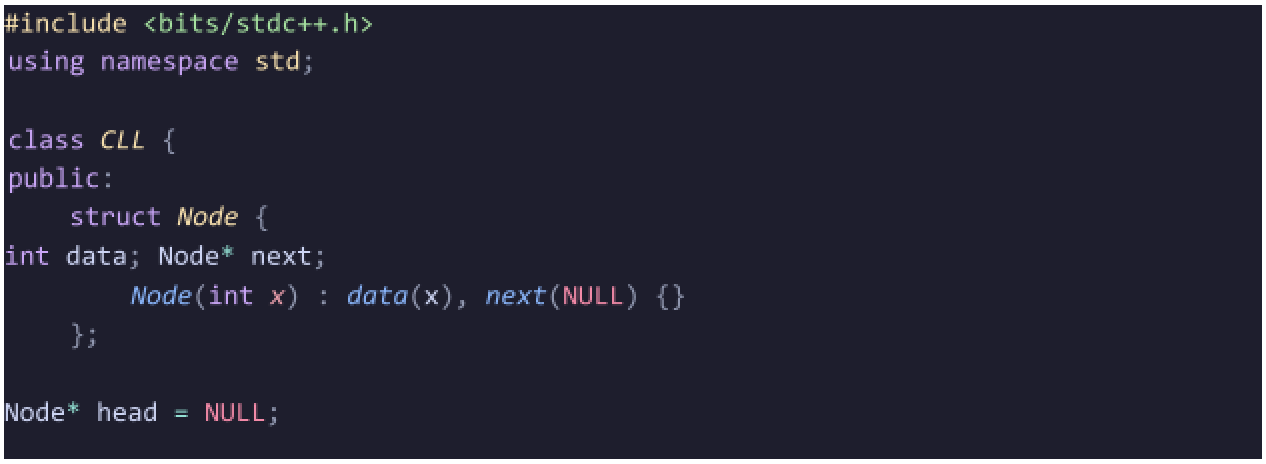
```
    c.insert(30);
    c.insert(40);

    c.deleteNode(30); // middle
    c.deleteNode(10); // head
    c.deleteNode(40); // last

    c.display();
}
```



```
#include <bits/stdc++.h>
using namespace std;

class CLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
Node* head = NULL; void

    insert(int x) {

        Node* n = new Node(x);
if(!head) {
head = n;
n->next = head; return;
        }
Node* t = head;
while(t->next != head) t = t->next; t->next
        = n;
n->next = head;
    }

    bool search(int key) {
        if(!head) return false;
```



```
Node* t = head; do {
            if(t->data == key)
                return true;

t = t->next;
} while(t != head); return
        false;
    }
};

int main() {
CLL cll;

    cll.insert(10);
    cll.insert(20);
    cll.insert(30);
int key = 25;

    if(cll.search(key))

cout << "Node " << key << " found\n";
else
cout << "Node " << key << " not found\n";
}
```



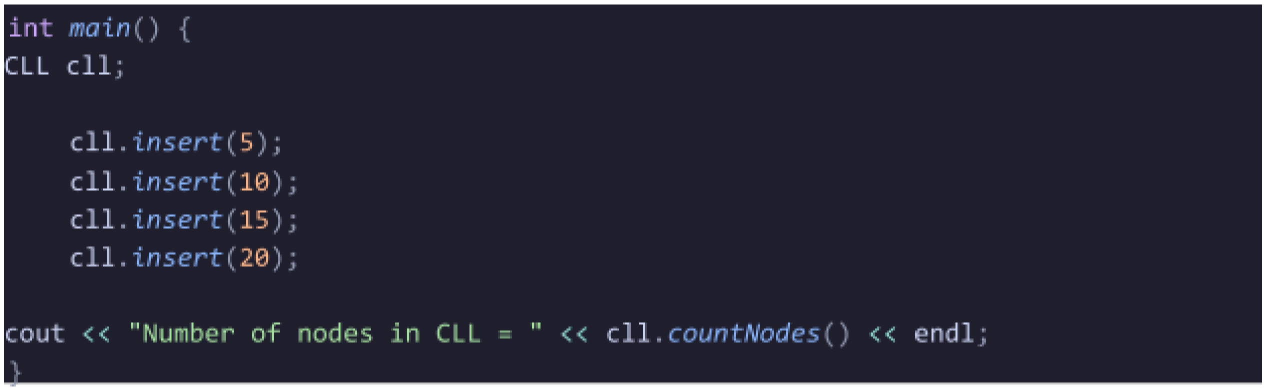
```
#include <bits/stdc++.h>
using namespace std;

class CLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };

Node* head = NULL;
```

```
    void insert(int x) {
        Node* n = new Node(x);
        if(!head) {
            head = n;
            n->next = head;
            return;
        }

        Node* t = head;
        while(t->next != head) t = t->next;

        t->next = n;
        n->next = head;
    }

    void displayWithHeadTwice() {
        if(!head) return;

        Node* t = head;

        // Print entire circular list
        do {
            cout << t->data << " ";
            t = t->next;
        } while(t != head);

        // Print head again
        cout << head->data << endl;
    }
};

int main() {
    CLL cll;

    cll.insert(10)
    ;
    cll.insert(20)
    ;
    cll.insert(30)
    ;

    cll.displayWithHeadTwice();
}
```

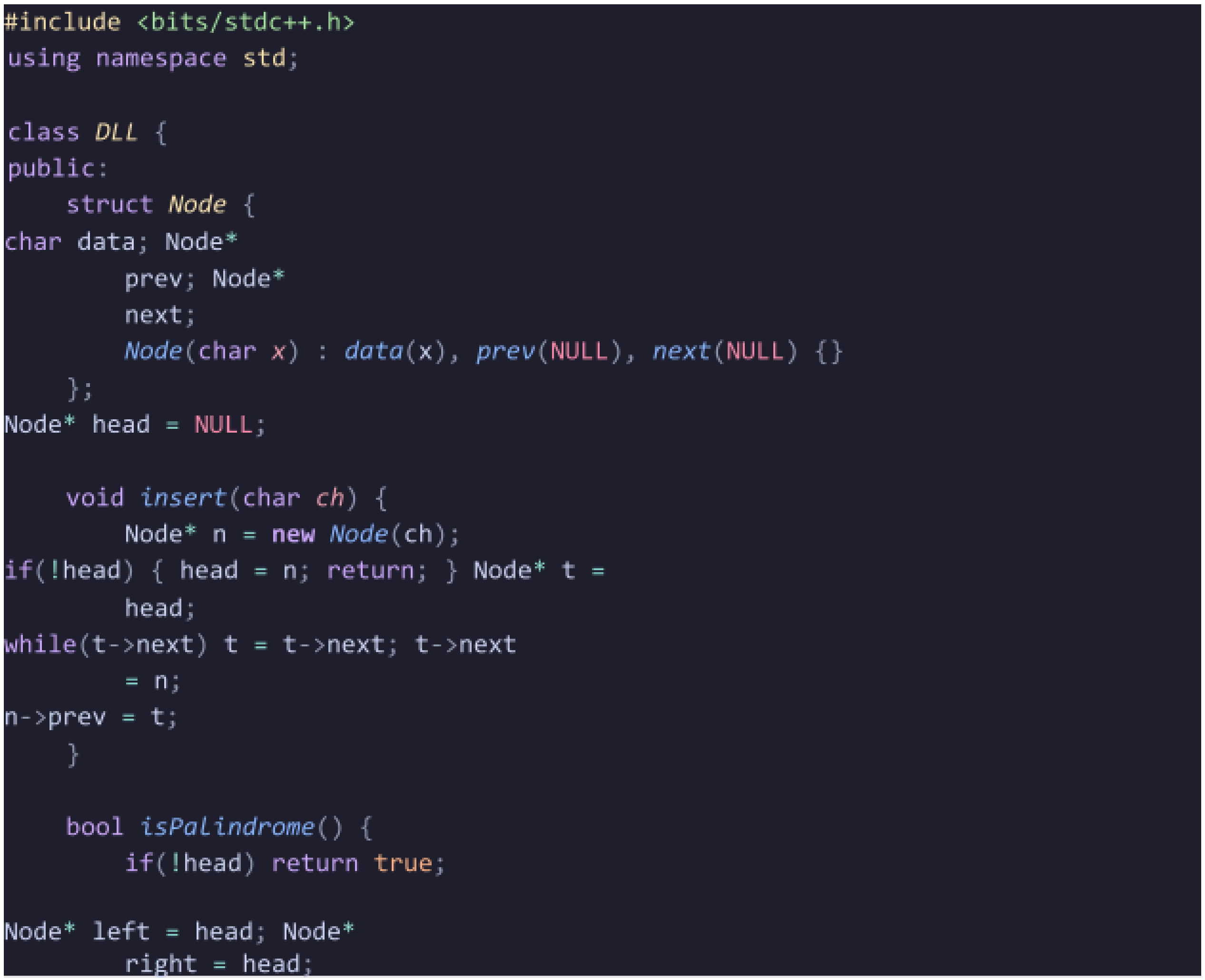
```
#include <bits/stdc++.h>
using namespace std;
```

```cpp
class DLL {
public:
    struct Node {
        int data;
        Node* prev;
        Node* next;
        Node(int x) : data(x), prev(NULL), next(NULL) {}
    };

    Node* head = NULL;

    void insert(int x) {
        Node* n = new Node(x);
        if(!head) {
            head = n;
            return;
        }
        Node* t = head;
        while(t->next) t = t->next;
        t->next = n;
        n->prev = t;
    }

    int countNodes() {
        int count = 0;
        Node* t = head;
        while(t) {
            count++;
            t = t->next;
        }
        return count;
    }
};

int main() {
    DLL dll;

    dll.insert(10)
    ;
    dll.insert(20)
    ;
    dll.insert(30)
    ;
    dll.insert(40)
    ;

    cout << "Number of nodes in DLL = " << dll.countNodes() << endl;
}
```

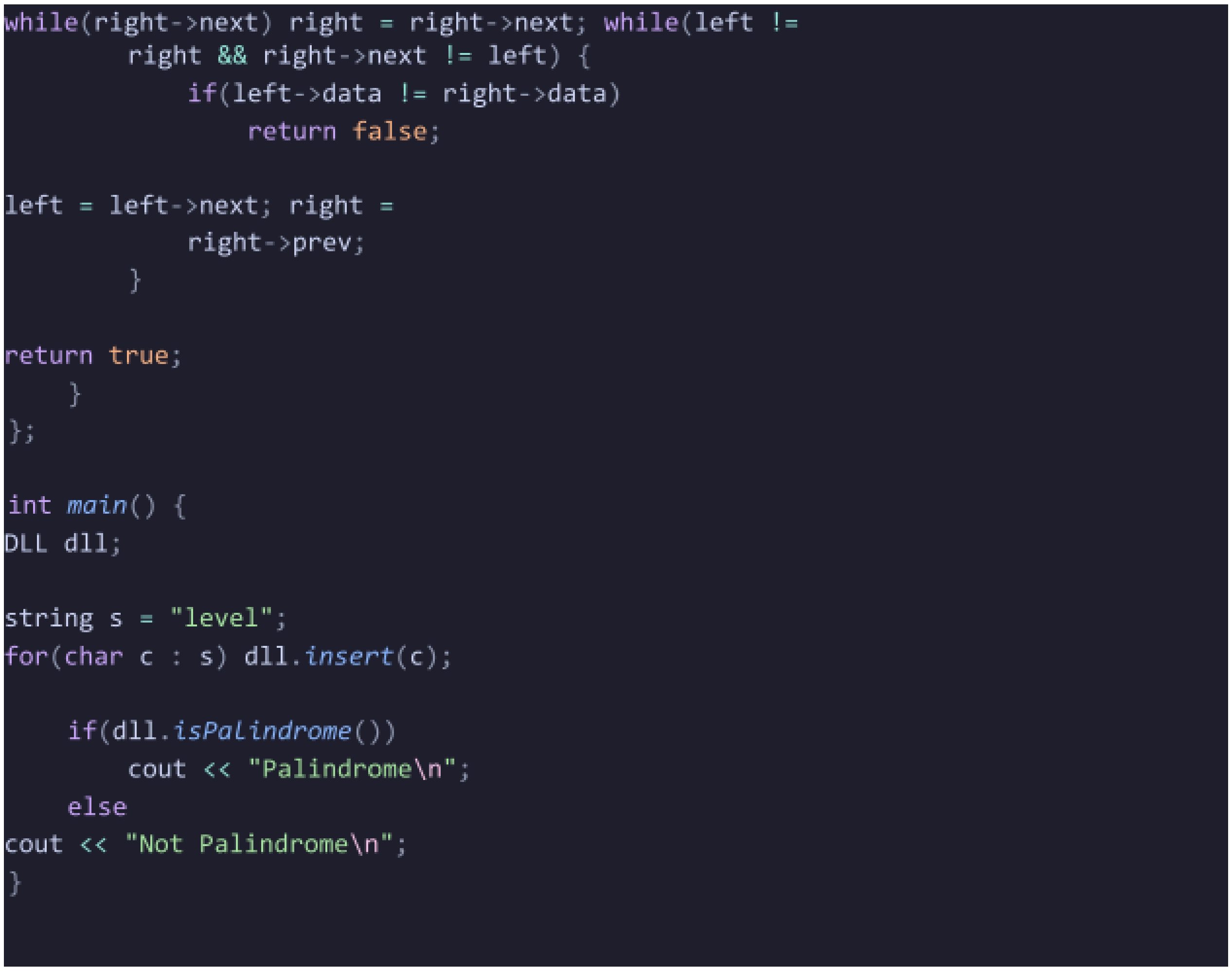
```cpp
#include <bits/stdc++.h>
using namespace std;

class CLL {
public:
    struct Node {
        int data;
        Node* next;
        Node(int x) : data(x), next(NULL) {}
    };

    Node* head = NULL;

    void insert(int x) {
        Node* n = new Node(x);

        if(!head) {
            head = n;
            n->next = head;
            return;
        }

        Node* t = head;
        while(t->next != head)
            t = t->next;

        t->next = n;
        n->next = head;
    }

    int countNodes() {
        if(!head) return 0;

        int count = 0;
        Node* t = head;

        do {
            count++;
            t = t->next;
        } while(t != head);

        return count;
    }
};
```



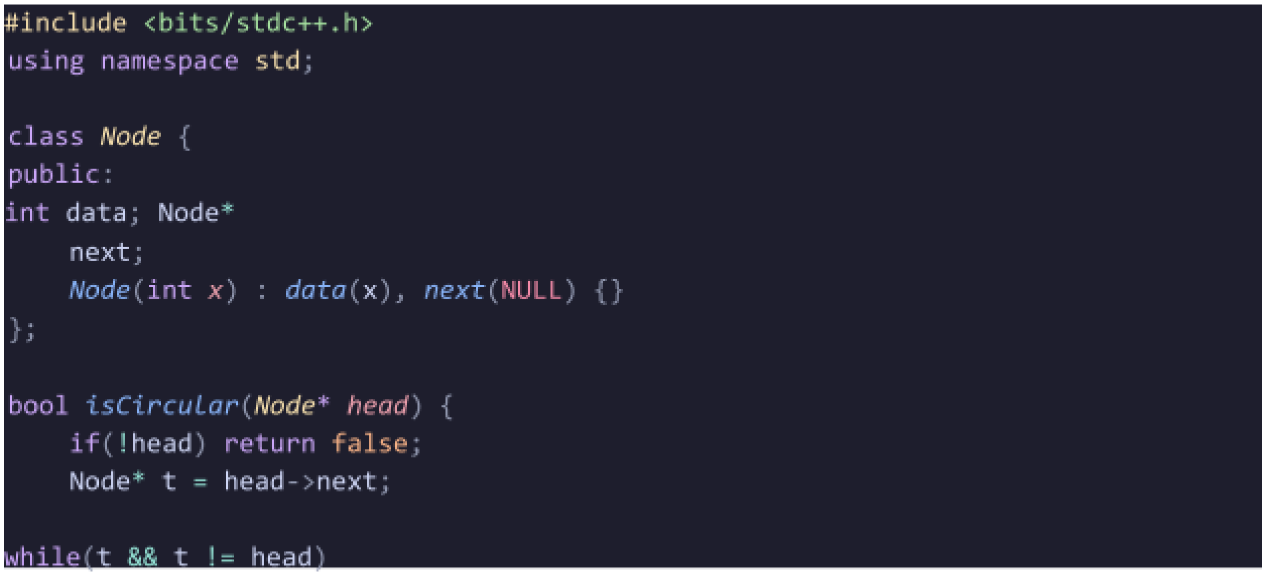
```
 int main() {
CLL cll;

    cll.insert(5);
    cll.insert(10);
    cll.insert(15);
    cll.insert(20);

cout << "Number of nodes in CLL = " << cll.countNodes() << endl;
}
```



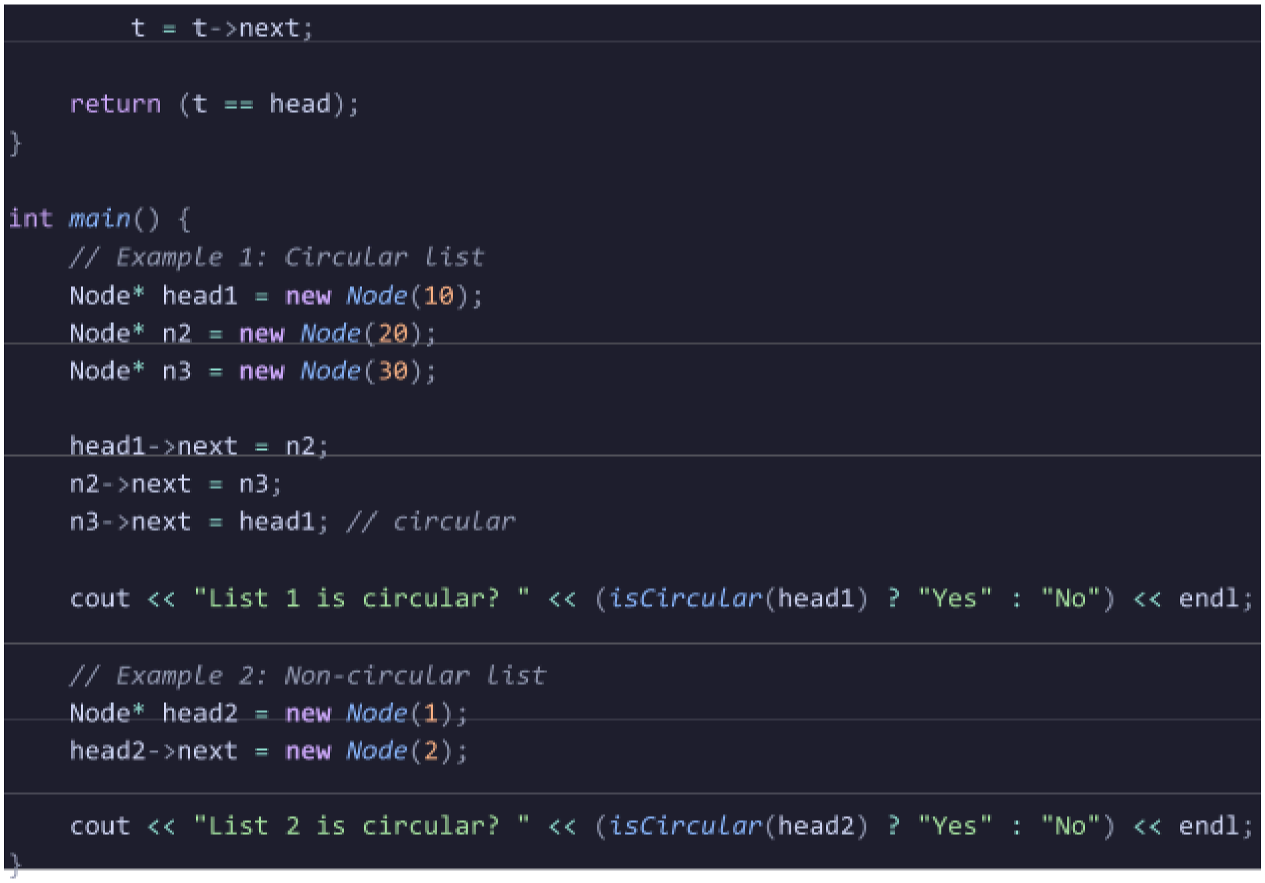
```
#include <bits/stdc++.h>
using namespace std;

class DLL {
public:
    struct Node {
char data; Node*
        prev; Node*
        next;
        Node(char x) : data(x), prev(NULL), next(NULL) {}
    };
Node* head = NULL;

    void insert(char ch) {
        Node* n = new Node(ch);
if(!head) { head = n; return; } Node* t =
        head;
while(t->next) t = t->next; t->next
        = n;
n->prev = t;
    }

    bool isPalindrome() {
        if(!head) return true;

Node* left = head; Node*
        right = head;
```



```
while(right->next) right = right->next; while(left !=
        right && right->next != left) {
            if(left->data != right->data)
                return false;

left = left->next; right =
            right->prev;
        }

return true;
    }
};

 int main() {
DLL dll;

string s = "level";
for(char c : s) dll.insert(c);

    if(dll.isPalindrome())
        cout << "Palindrome\n";
    else
cout << "Not Palindrome\n";
 }
```



```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
int data; Node*
    next;
    Node(int x) : data(x), next(NULL) {}
};

bool isCircular(Node* head) {
    if(!head) return false;
    Node* t = head->next;

while(t && t != head)
```



```cpp
        t = t->next;

    return (t == head);
}

int main() {
    // Example 1: Circular list
    Node* head1 = new Node(10);
    Node* n2 = new Node(20);
    Node* n3 = new Node(30);

    head1->next = n2;
    n2->next = n3;
    n3->next = head1; // circular

    cout << "List 1 is circular? " << (isCircular(head1) ? "Yes" : "No") << endl;

    // Example 2: Non-circular list
    Node* head2 = new Node(1);
    head2->next = new Node(2);

    cout << "List 2 is circular? " << (isCircular(head2) ? "Yes" : "No") << endl;
}
```