Ques 1

```cpp
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* prev;
    Node* next;
};

Node* head = nullptr;

void insertFirst(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->prev = nullptr;
    newNode->next = head;
    if(head) head->prev = newNode;
    head = newNode;
}

void insertLast(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;
    if(!head) {
        newNode->prev = nullptr;
        head = newNode;
        return;
    }
    Node* temp = head;
    while(temp->next) temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

void insertAfter(int key, int value) {
    Node* temp = head;
    while(temp && temp->data != key) temp = temp->next;
    if(!temp) {
        cout << "Node " << key << " not found\n";
        return;
    }
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = temp->next;
    newNode->prev = temp;
    if(temp->next) temp->next->prev = newNode;
    temp->next = newNode;
}

void deleteNode(int key) {
    Node* temp = head;
    while(temp && temp->data != key) temp = temp->next;
    if(!temp) {
        cout << "Node " << key << " not found\n";
        return;
    }
    if(temp->prev) temp->prev->next = temp->next;
    else head = temp->next;
    if(temp->next) temp->next->prev = temp->prev;
    delete temp;
    cout << "Node " << key << " deleted\n";
}

void searchNode(int key) {
    Node* temp = head;
    int pos = 1;
    while(temp) {
        if(temp->data == key) {
            cout << "Node " << key << " found at position " << pos << "\n";
            return;
        }
        temp = temp->next;
        pos++;
    }
    cout << "Node " << key << " not found\n";
}

void display() {
    Node* temp = head;
    cout << "List: ";
    while(temp) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << "\n";
}
```

```
int main() {
    int choice, value, key;
    while(true) {
        cout << "\n1.Insert First  2.Insert Last  3.Insert After  4.Delete Node  5.Search Node
6.Display  7.Exit\n";
        cout << "Enter choice: ";
        cin >> choice;
        switch(choice) {
            case 1:
                cout << "Enter value: ";
                cin >> value;
                insertFirst(value);
                break;
            case 2:
                cout << "Enter value: ";
                cin >> value;
                insertLast(value);
                break;
            case 3:
                cout << "Enter node after which to insert: ";
                cin >> key;
                cout << "Enter value: ";
                cin >> value;
                insertAfter(key, value);
                break;
            case 4:
                cout << "Enter node to delete: ";
                cin >> key;
                deleteNode(key);
                break;
            case 5:
                cout << "Enter node to search: ";
                cin >> key;
                searchNode(key);
                break;
            case 6:
                display();
                break;
            case 7:
                return 0;
            default:
                cout << "Invalid choice\n";
        }
    }
}
```

Ques 2

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

Node* head = nullptr;

void insertLast(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    if(!head) {
        head = newNode;
        newNode->next = head;
        return;
    }
    Node* temp = head;
    while(temp->next != head) temp = temp->next;
    temp->next = newNode;
    newNode->next = head;
}

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

int main() {
    insertLast(20);
    insertLast(100);
    insertLast(40);
    insertLast(80);
    insertLast(60);
    display();
    return 0;
}
```

QUES 3

```
#include <iostream>
using namespace std;

// Doubly Linked List
struct DNode {
    int data;
    DNode* prev;
    DNode* next;
};

DNode* dHead = nullptr;

void insertDLL(int value) {
    DNode* newNode = new DNode();
    newNode->data = value;
    newNode->prev = nullptr;
    newNode->next = dHead;
    if(dHead) dHead->prev = newNode;
    dHead = newNode;
}

int sizeDLL() {
    int count = 0;
    DNode* temp = dHead;
    while(temp) {
        count++;
        temp = temp->next;
    }
    return count;
}

// Circular Linked List
struct CNode {
    int data;
    CNode* next;
};

CNode* cHead = nullptr;
```

```
void insertCLL(int value) {
    CNode* newNode = new CNode();
    newNode->data = value;
    if(!cHead) {
        cHead = newNode;
        newNode->next = cHead;
        return;
    }
    CNode* temp = cHead;
    while(temp->next != cHead) temp = temp->next;
    temp->next = newNode;
    newNode->next = cHead;
}

int sizeCLL() {
    if(!cHead) return 0;
    int count = 1;
    CNode* temp = cHead;
    while(temp->next != cHead) {
        count++;
        temp = temp->next;
    }
    return count;
}

int main() {

    insertDLL(10);
    insertDLL(20);
    insertDLL(30);
    cout << "Size of Doubly Linked List: " << sizeDLL() << endl;


    insertCLL(100);
    insertCLL(200);
    insertCLL(300);
    insertCLL(400);
    cout << "Size of Circular Linked List: " << sizeCLL() << endl;

    return 0;
}
```
QUES 4
```
#include <iostream>
```

```
using namespace std;

struct Node {
    char data;
    Node* prev;
    Node* next;
};

Node* head = nullptr;

void insertLast(char value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;
    if(!head) {
        newNode->prev = nullptr;
        head = newNode;
        return;
    }
    Node* temp = head;
    while(temp->next) temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

bool isPalindrome() {
    if(!head) return true;
    Node* left = head;
    Node* right = head;
    while(right->next) right = right->next;
    while(left != right && left->prev != right) {
        if(left->data != right->data) return false;
        left = left->next;
        right = right->prev;
    }
    return true;
}

void display() {
    Node* temp = head;
    while(temp) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    insertLast('L');
    insertLast('E');
    insertLast('V');
    insertLast('E');
    insertLast('L');

    cout << "Doubly Linked List: ";
    display();

    if(isPalindrome()) cout << "The list is a palindrome\n";
    else cout << "The list is not a palindrome\n";

    return 0;
}
```

QUES 5

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

Node* head = nullptr;

void insertLast(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;
    if(!head) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while(temp->next) temp = temp->next;
    temp->next = newNode;
}

bool isCircular() {
    if(!head) return true;
    Node* temp = head->next;
    while(temp && temp != head) temp = temp->next;
    return (temp == head);
}

int main() {
    insertLast(2);
    insertLast(4);
    insertLast(6);
    insertLast(7);
    insertLast(5);

    if(isCircular()) cout << "The linked list is circular\n";
    else cout << "The linked list is not circular\n";

    return 0;
}
```