Q1

```
#include <iostream>

using namespace std;

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

Node* last = NULL;

void insertFirst() {
    int val;
    cout << "Enter value: ";
    cin >> val;

    Node* newNode = new Node();
    newNode->data = val;

    if (last == NULL) {
        last = newNode;
        newNode->next = last;
    } else {
        newNode->next = last->next;
        last->next = newNode;
    }
    cout << "Node insertd.\n";
}

void insertLast() {
    int val;
    cout << "Enter value: ";
    cin >> val;

    Node* newNode = new Node();
    newNode->data = val;

    if (last == NULL) {
        last = newNode;
        newNode->next = last;
    } else {
        newNode->next = last->next;
```

```
        last->next = newNode;
        last = newNode;
    }
    cout << "Node insertd.\n";
}

void insertAfter() {
    int key, val;
    cout << "Enter value to insert: ";
    cin >> val;
    cout << "Enter value to insert after: ";
    cin >> key;

    if (last == NULL) {
        cout << "List is emty.\n";
        return;
    }

    Node* temp = last->next;

    do {
        if (temp->data == key) {
            Node* newNode = new Node();
            newNode->data = val;
            newNode->next = temp->next;
            temp->next = newNode;

            if (temp == last) {
                last = newNode;
            }
            cout << "Node insertd.\n";
            return;
        }
        temp = temp->next;
    } while (temp != last->next);

    cout << "Node not found.\n";
}

void deleteNode() {
    int key;
    cout << "Enter value to delete: ";
    cin >> key;
```

```
    if (last == NULL) {
        cout << "List is emty.\n";
        return;
    }

    Node* curr = last->next;
    Node* prev = last;

    do {
        if (curr->data == key) {
            if (curr == last && curr->next == last) {
                last = NULL;
                delete curr;
            } else if (curr == last->next) {
                last->next = curr->next;
                delete curr;
            } else if (curr == last) {
                prev->next = last->next;
                last = prev;
                delete curr;
            } else {
                prev->next = curr->next;
                delete curr;
            }
            cout << "Node deletd.\n";
            return;
        }
        prev = curr;
        curr = curr->next;
    } while (curr != last->next);

    cout << "Node not found.\n";
}

void searchNode() {
    int key;
    cout << "Enter value to serch: ";
    cin >> key;

    if (last == NULL) {
        cout << "List is emty.\n";
        return;
    }
```

```
    Node* temp = last->next;
    int pos = 0;

    do {
        if (temp->data == key) {
            cout << "Node found at positon " << pos << endl;
            return;
        }
        temp = temp->next;
        pos++;
    } while (temp != last->next);

    cout << "Node not found.\n";
}

void displayList() {
    if (last == NULL) {
        cout << "List is emty.\n";
        return;
    }

    Node* temp = last->next;
    cout << "List: ";

    do {
        cout << temp->data << " -> ";
        temp = temp->next;
    } while (temp != last->next);

    cout << "(back to " << last->next->data << ")\n";
}

int main() {
    int ch;
    while (1) {
        cout << "\n-- Circular Linkd List Menu --\n";
        cout << "1. Insert First\n";
        cout << "2. Insert Last\n";
        cout << "3. Insert After\n";
        cout << "4. Delete Node\n";
        cout << "5. Search Node\n";
        cout << "6. Display List\n";
        cout << "7. Exit\n";
        cout << "Enter your choise: ";
```

```
        cin >> ch;

        switch (ch) {
            case 1: insertFirst(); break;
            case 2: insertLast(); break;
            case 3: insertAfter(); break;
            case 4: deleteNode(); break;
            case 5: searchNode(); break;
            case 6: displayList(); break;
            case 7: return 0;
            default: cout << "Invalid choise.\n";
        }
    }
    return 0;
}
```

```
-- Circular Linkd List Menu --
1. Insert First
2. Insert Last
3. Insert After
4. Delete Node
5. Search Node
6. Display List
7. Exit
Enter your choise: 1
Enter value: 2
Node insertd.

-- Circular Linkd List Menu --
1. Insert First
2. Insert Last
3. Insert After
4. Delete Node
5. Search Node
6. Display List
7. Exit
Enter your choise: 2
Enter value: 4
Node insertd.

-- Circular Linkd List Menu --
1. Insert First
2. Insert Last
3. Insert After
4. Delete Node
5. Search Node
6. Display List
```

```
Enter value to insert: 4
Enter value to insert after: 5
Node not found.

-- Circular Linkd List Menu --
1. Insert First
2. Insert Last
3. Insert After
4. Delete Node
5. Search Node
6. Display List
7. Exit
Enter your choise: 5
Enter value to serch: 6
Node not found.

-- Circular Linkd List Menu --
1. Insert First
2. Insert Last
3. Insert After
4. Delete Node
5. Search Node
6. Display List
7. Exit
Enter your choise: 7


...Program finished with exit code 0
Press ENTER to exit console.
```

Q2

```
#include <iostream>

using namespace std;
```

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

int main() {
    Node* head = NULL;
    Node* n1 = new Node();
    Node* n2 = new Node();
    Node* n3 = new Node();
    Node* n4 = new Node();
    Node* n5 = new Node();

    n1->data = 20;
    n2->data = 100;
    n3->data = 40;
    n4->data = 80;
    n5->data = 60;

    head = n1;
    n1->next = n2;
    n2->next = n3;
    n3->next = n4;
    n4->next = n5;
    n5->next = head;


    if (head == NULL) {
        cout << "List is emty.\n";
        return 0;
    }

    Node* temp = head;

    cout << "Output: ";
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);


    cout << head->data << endl;

    return 0;
```

}

```
Output: 20 100 40 80 60 20


...Program finished with exit code 0
Press ENTER to exit console.
```

Q3

```
#include <iostream>

using namespace std;


struct DNode {
    int data;
    DNode *prev, *next;
};

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


int getDLLSize(DNode* head) {
    int count = 0;
    DNode* temp = head;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }
    return count;
}
```

```
int getCLLSize(CNode* last) {
    if (last == NULL) {
        return 0;
    }

    int count = 0;
    CNode* temp = last->next;

    do {
        count++;
        temp = temp->next;
    } while (temp != last->next);

    return count;
}

int main() {

    DNode* dhead = new DNode();
    DNode* d2 = new DNode();
    dhead->data = 10;
    dhead->prev = NULL;
    dhead->next = d2;
    d2->data = 20;
    d2->prev = dhead;
    d2->next = NULL;

    cout << "Size of Doubly Linkd List: " << getDLLSize(dhead) << endl;


    CNode* clast = new CNode();
    CNode* c2 = new CNode();
    clast->data = 10;
    clast->next = c2;
    c2->data = 20;
    c2->next = clast;


    cout << "Size of Circular Linkd List: " << getCLLSize(c2) << endl;

    return 0;
}
```

```
Size of Doubly Linkd List: 2
Size of Circular Linkd List: 2
```

Q4

```
#include <iostream>

using namespace std;

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

bool isPalindrome(Node* head) {
    if (head == NULL) {
        return true;
    }


    Node* tail = head;
    while (tail->next != NULL) {
        tail = tail->next;
    }

    Node* left = head;
    Node* right = tail;

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

    return true;
}
```

```
int main() {

    Node* head = new Node();
    Node* n2 = new Node();
    Node* n3 = new Node();
    Node* n4 = new Node();
    Node* n5 = new Node();

    head->data = 'L';
    head->prev = NULL;
    head->next = n2;

    n2->data = 'E';
    n2->prev = head;
    n2->next = n3;

    n3->data = 'V';
    n3->prev = n2;
    n3->next = n4;

    n4->data = 'E';
    n4->prev = n3;
    n4->next = n5;

    n5->data = 'L';
    n5->prev = n4;
    n5->next = NULL;

    if (isPalindrome(head)) {
        cout << "Output: True\n";
    } else {
        cout << "Output: False\n";
    }


    n4->data = 'X';

    if (isPalindrome(head)) {
        cout << "Output (test 2): True\n";
    } else {
        cout << "Output (test 2): False\n";
    }

    return 0;
```

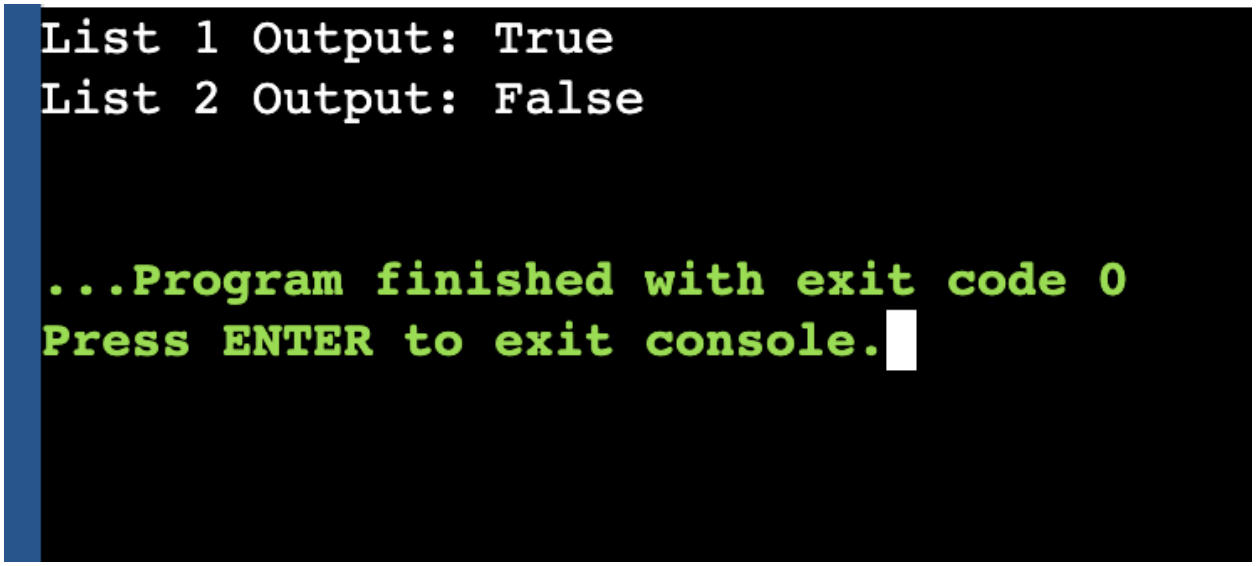
}

```
Output: True
Output (test 2): False


...Program finished with exit code 0
Press ENTER to exit console.
```

Q5

```
#include <iostream>

using namespace std;

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


bool isCircular(Node* head) {
    if (head == NULL) {
        return false;
    }

    Node* slow = head;
    Node* fast = head;

    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;

        if (slow == fast) {
            return true;
        }
    }
```

```
    return false;
}

int main() {

    Node* head1 = new Node();
    Node* n2 = new Node();
    Node* n3 = new Node();
    head1->data = 2;
    head1->next = n2;
    n2->data = 4;
    n2->next = n3;
    n3->data = 6;
    n3->next = head1;

    if (isCircular(head1)) {
        cout << "List 1 Output: True\n";
    } else {
        cout << "List 1 Output: False\n";
    }


    Node* head2 = new Node();
    Node* n4 = new Node();
    head2->data = 1;
    head2->next = n4;
    n4->data = 2;
    n4->next = NULL;

    if (isCircular(head2)) {
        cout << "List 2 Output: True\n";
    } else {
        cout << "List 2 Output: False\n";
    }

    return 0;
}
```


List 1 Output: True
List 2 Output: False
...Program finished with exit code 0
Press ENTER to exit console.