DSA Lab Assignment 6

Aarnav Dhillon

1024030379

Group 2C25

Q1A

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

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

Node* head = NULL;
Node* tail = NULL;
Node* temp = NULL;
int n = 0;

void createList() {
    cout << "Enter the number of nodes you want: ";
    cin >> n;

    if (n < 1) {
        cout << "Enter a valid value" << endl;
        return;
    }

    cout << "Enter the values of your nodes:" << endl;
```

```
    for (int i = 0; i < n; i++) {
        Node* newNode = new Node();
        cin >> newNode->val;

        if (head == NULL) {
            head = newNode;
            tail = newNode;
            newNode->next = head;
        } else {
            tail->next = newNode;
            newNode->next = head;
            tail = newNode;
        }
    }
}

void insertNode() {
    cout << "Enter the position where you want to insert your node: ";
    int x;
    cin >> x;

    if (x < 1 || x > n + 1) {
        cout << "Enter a valid position" << endl;
        return;
    }

    Node* newNode = new Node();
    cout << "Enter the value of your node: ";
```

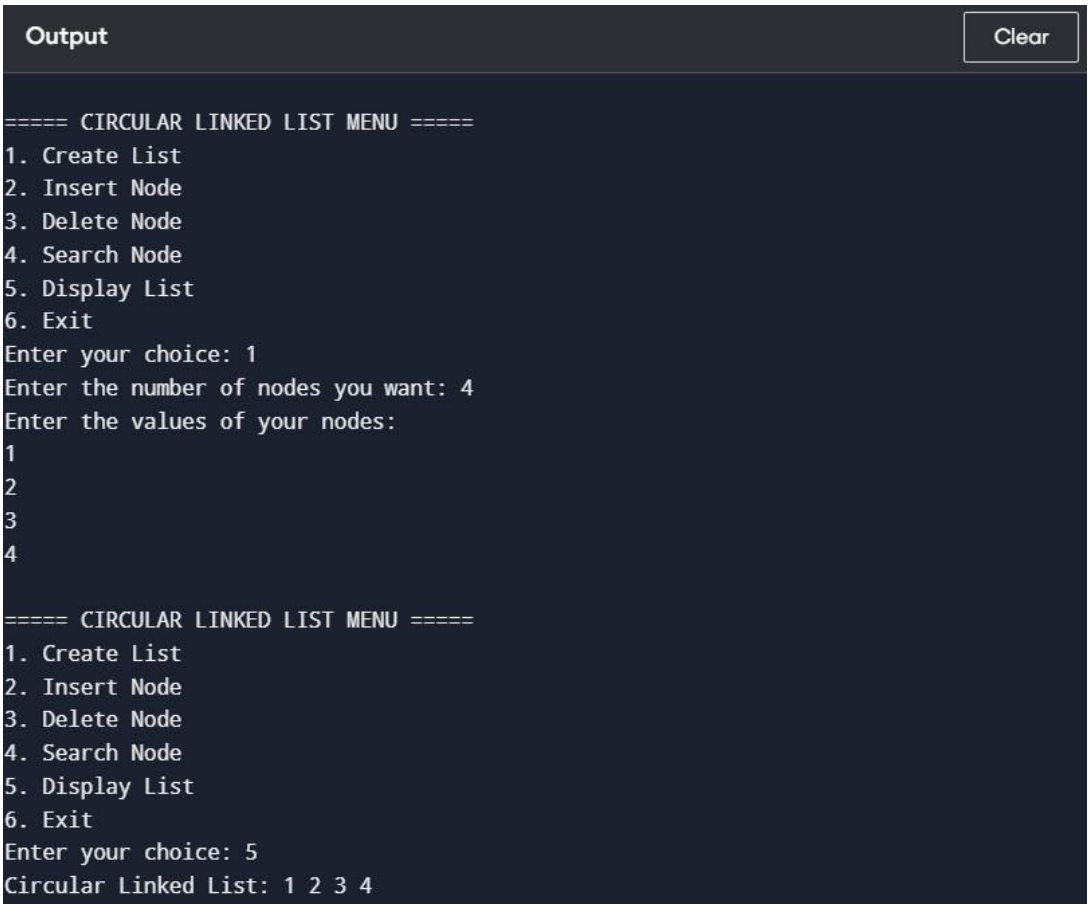
```
    cin >> newNode->val;

    if (x == 1) {
        newNode->next = head;
        tail->next = newNode;
        head = newNode;
    } else if (x == n + 1) {
        tail->next = newNode;
        newNode->next = head;
        tail = newNode;
    } else {
        temp = head;
        for (int i = 1; i < x - 1; i++) {
            temp = temp->next;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
    n++;
}

void deleteNode() {
    cout << "Enter the node position you want to delete: ";
    int x;
    cin >> x;

    if (x < 1 || x > n) {
        cout << "Enter a valid value" << endl;
```

```
    return;
}

if (x == 1) {
    Node* tempNode = head;
    tail->next = head->next;
    head = head->next;
    delete tempNode;
} else if (x == n) {
    temp = head;
    while (temp->next != tail) {
        temp = temp->next;
    }
    Node* tempNode = tail;
    temp->next = head;
    tail = temp;
    delete tempNode;
} else {
    temp = head;
    for (int i = 1; i < x - 1; i++) {
        temp = temp->next;
    }
    Node* tempNode = temp->next;
    temp->next = temp->next->next;
    delete tempNode;
}

n--;
```

```
}

void searchNode() {
    cout << "Enter the value you want to search: ";
    int x;
    cin >> x;
    temp = head;
    bool found = false;

    if (head == NULL) {
        cout << "List is empty" << endl;
        return;
    }

    do {
        if (temp->val == x) {
            found = true;
            break;
        }
        temp = temp->next;
    } while (temp != head);

    if (found)
        cout << "Found!" << endl;
    else
        cout << "Not found" << endl;
}
```

```
void display() {
    if (head == NULL) {
        cout << "List is empty" << endl;
        return;
    }

    temp = head;
    cout << "Circular Linked List: ";
    do {
        cout << temp->val << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}

int main() {
    int choice;
    while (true) {
        cout << "\n===== CIRCULAR LINKED LIST MENU =====\n";
        cout << "1. Create List\n";
        cout << "2. Insert Node\n";
        cout << "3. Delete Node\n";
        cout << "4. Search Node\n";
        cout << "5. Display List\n";
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
```

```
        switch (choice) {
            case 1:
                createList();
                break;
            case 2:
                insertNode();
                break;
            case 3:
                deleteNode();
                break;
            case 4:
                searchNode();
                break;
            case 5:
                display();
                break;
            case 6:
                cout << "Exiting program..." << endl;
                return 0;
            default:
                cout << "Enter a valid choice." << endl;
        }
    }
}
```



Q1B

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

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

Node* head = NULL;
int n = 0;
```

```
void createList() {
    cout << "Enter the number of nodes you want: ";
    cin >> n;

    if (n < 1) {
        cout << "Enter a valid value" << endl;
        return;
    }

    head = NULL;
    Node* temp = NULL;

    cout << "Enter the values of your nodes:" << endl;
    for (int i = 0; i < n; i++) {
        Node* newNode = new Node();
        cin >> newNode->val;
        newNode->next = NULL;
        newNode->prev = NULL;

        if (head == NULL) {
            head = newNode;
            temp = head;
        } else {
            temp->next = newNode;
            newNode->prev = temp;
            temp = newNode;
        }
```

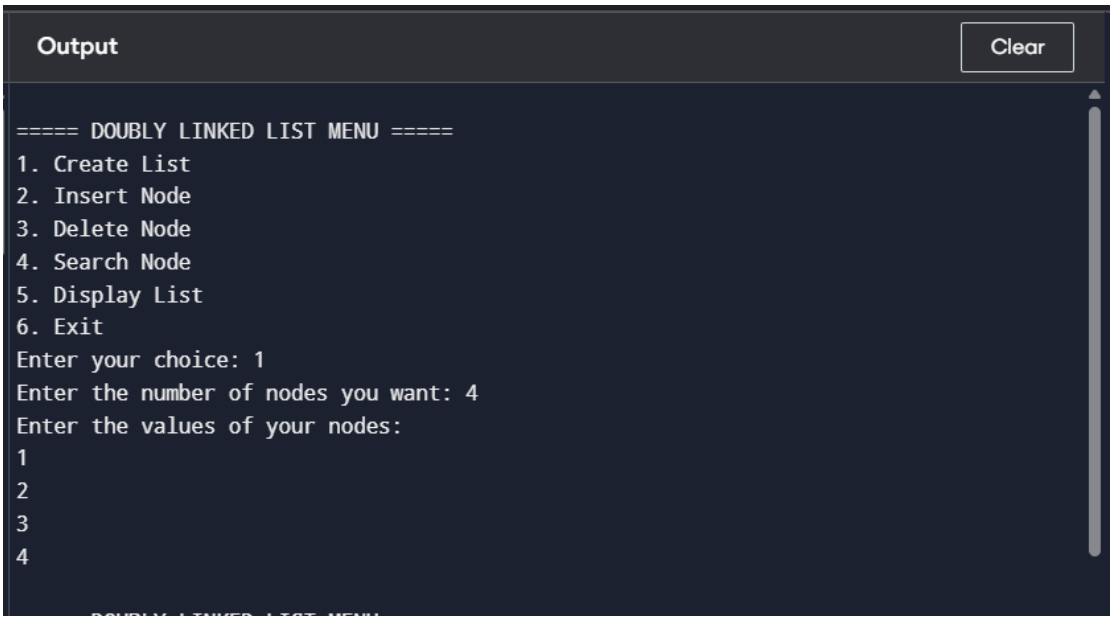
```
    }
}

void insertNode() {
    cout << "Enter the position where you want to insert your node: ";
    int x;
    cin >> x;

    if (x < 1 || x > n + 1) {
        cout << "Enter a valid position" << endl;
        return;
    }

    Node* newNode = new Node();
    cout << "Enter the value of your node: ";
    cin >> newNode->val;
    newNode->next = NULL;
    newNode->prev = NULL;

    if (head == NULL) {
        head = newNode;
    }
    else if (x == 1) {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    }
    else {
```

```
        Node* temp = head;
        for (int i = 1; i < x - 1 && temp->next != NULL; i++) {
            temp = temp->next;
        }
        newNode->next = temp->next;
        newNode->prev = temp;
        if (temp->next != NULL) temp->next->prev = newNode;
        temp->next = newNode;
    }
    n++;
}

void deleteNode() {
    cout << "Enter the node position you want to delete: ";
    int x;
    cin >> x;

    if (x < 1 || x > n || head == NULL) {
        cout << "Enter a valid value" << endl;
        return;
    }

    if (x == 1) {
        Node* tempNode = head;
        head = head->next;
        if (head != NULL) head->prev = NULL;
        delete tempNode;
    }
```

```
    else {
        Node* temp = head;
        for (int i = 1; i < x; i++) temp = temp->next;
        Node* tempNode = temp;
        if (temp->next != NULL) temp->next->prev = temp->prev;
        if (temp->prev != NULL) temp->prev->next = temp->next;
        delete tempNode;
    }
    n--;
}

void searchNode() {
    cout << "Enter the value you want to search: ";
    int x;
    cin >> x;
    Node* temp = head;
    bool found = false;

    while (temp != NULL) {
      if (temp->val == x) {
            found = true;
            break;
        }
        temp = temp->next;
    }

    if (found)
        cout << "Found!" << endl;
```

```
    else
        cout << "Not found" << endl;
}

void display() {
    if (head == NULL) {
        cout << "List is empty" << endl;
        return;
    }

    Node* temp = head;
    cout << "Doubly Linked List: ";
    while (temp != NULL) {
        cout << temp->val << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    int choice;
    while (true) {
        cout << "\n===== DOUBLY LINKED LIST MENU =====\n";
        cout << "1. Create List\n";
        cout << "2. Insert Node\n";
        cout << "3. Delete Node\n";
        cout << "4. Search Node\n";
        cout << "5. Display List\n";
```

```
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: createList(); break;
            case 2: insertNode(); break;
            case 3: deleteNode(); break;
            case 4: searchNode(); break;
            case 5: display(); break;
            case 6: cout << "Exiting program..." << endl; return 0;
            default: cout << "Enter a valid choice." << endl;
        }
    }
}
```



Q2

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

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

Node* head = NULL;
Node* tail = NULL;
Node* temp = NULL;
int n = 0;

void createList() {
   cout << "Enter the number of nodes you want: ";
   cin >> n;

   if (n < 1) {
      cout << "Enter a valid value" << endl;
      return;
   }

   cout << "Enter the values of your nodes:" << endl;
   for (int i = 0; i < n; i++) {
      Node* newNode = new Node();
      cin >> newNode->val;
```

```
        if (head == NULL) {
            head = newNode;
            tail = newNode;
            newNode->next = head;  // circular link
        } else {
            tail->next = newNode;
            newNode->next = head;
            tail = newNode;
        }
    }
}



void display() {
    if (head == NULL) {
        cout << "List is empty" << endl;
        return;
    }

    temp = head;
    cout << "Circular Linked List: ";
    do {
        cout << temp->val << " ";
        temp = temp->next;
    } while (temp != head);
    cout<<temp->val;
    cout << endl;
}
```

```
int main() {
   createList();
   display();

}
```

Output

```
Enter the number of nodes you want: 4
Enter the values of your nodes:
1
2
3
4
Circular Linked List: 1 2 3 4 1
```

Q3A

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

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

Node* head = NULL;
Node* tail = NULL;
Node* temp = NULL;
int n = 0;
```

```
void createList() {
    cout << "Enter the number of nodes you want: ";
    cin >> n;

    if (n < 1) {
        cout << "Enter a valid value" << endl;
        return;
    }

    cout << "Enter the values of your nodes:" << endl;
    for (int i = 0; i < n; i++) {
        Node* newNode = new Node();
        cin >> newNode->val;

        if (head == NULL) {
            head = newNode;
            tail = newNode;
            newNode->next = head;
        } else {
            tail->next = newNode;
            newNode->next = head;
            tail = newNode;
        }
    }
}

void size() {
    if (head == NULL) {
```

```
        cout << "List is empty" << endl;
        return;
    }

    int count = 0;
    temp = head;

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

    cout << "Number of nodes in Circular Linked List: " << count << endl;
    cout << "Approximate total memory used: " << count * sizeof(Node) << " bytes" << endl;
}

int main() {
    createList();
    size();
    return 0;
}
```

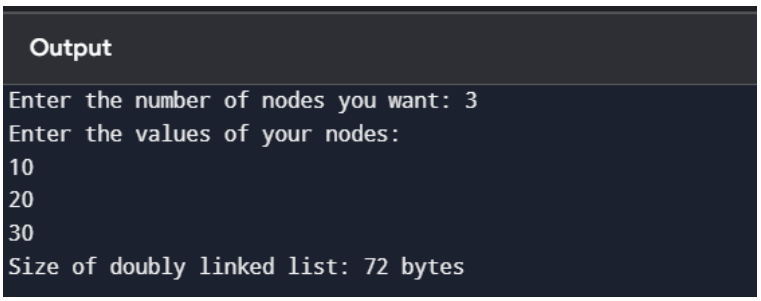
```
Enter the number of nodes you want: 3
Enter the values of your nodes:
10
20
30
Number of nodes in Circular Linked List: 3
Approximate total memory used: 48 bytes
```

Q3B

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

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

Node* head = NULL;
Node* tail = NULL;
Node* temp = NULL;
int n = 0;

void createList() {
   cout << "Enter the number of nodes you want: ";
   cin >> n;

   if (n < 1) {
      cout << "Enter a valid value" << endl;
      return;
   }

   cout << "Enter the values of your nodes:" << endl;
   for (int i = 0; i < n; i++) {
      Node* newNode = new Node();
      cin >> newNode->val;
```

```
        newNode->next = NULL;
        newNode->prev = NULL;

        if (head == NULL) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }
}

void size() {
    if (head == NULL) {
        cout << "List is empty" << endl;
        return;
    }

    int i = 0;
    temp = head;
    while (temp != NULL) {
        i++;
        temp = temp->next;
    }

    cout << "Size of doubly linked list: " << i * sizeof(Node) << " bytes" << endl;
}
```

```
int main() {
   createList();
   size();
}
```



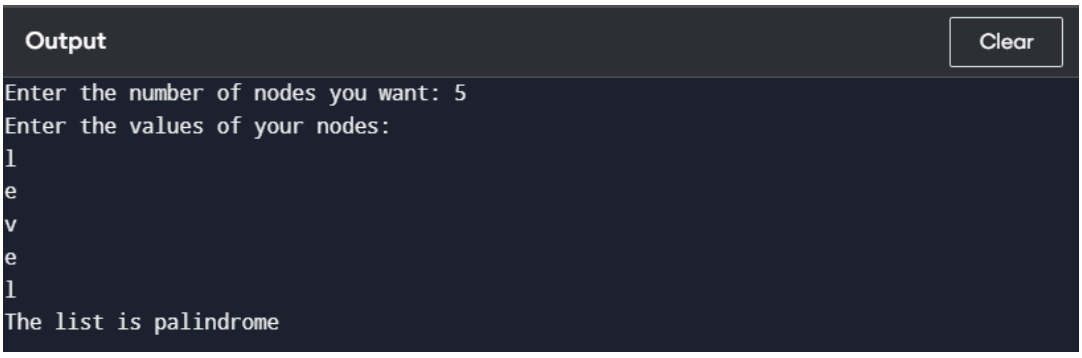
Q4

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

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

Node* head = NULL;
Node* tail = NULL;
Node* temp = NULL;
int n = 0;
```

```
void createList() {
    cout << "Enter the number of nodes you want: ";
    cin >> n;

    if (n < 1) {
        cout << "Enter a valid value" << endl;
        return;
    }

    cout << "Enter the values of your nodes:" << endl;
    for (int i = 0; i < n; i++) {
        Node* newNode = new Node();
        cin >> newNode->val;
        newNode->next = NULL;
        newNode->prev = NULL;

        if (head == NULL) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }
}

bool Palindrome() {
    if (head == NULL) {
```

```
      cout << "List is empty" << endl;
      return true;
  }


Node* fast=head;
Node*slow=head;
while(fast!=NULL&&fast->next!=NULL){
    fast=fast->next->next;
    slow=slow->next;
}
 temp=slow->prev;
if(n%2==0){

    while(temp!=NULL&&slow!=NULL){
    if(temp->val!=slow->val){
        return false;
    }
       temp=temp->prev;
       slow=slow->next;
    }
    return true;
}

    slow=slow->next;
      while(temp!=NULL&&slow!=NULL){
    if(temp->val!=slow->val){
        return false;
```

```
    }
      temp=temp->prev;
      slow=slow->next;
    }
    return true;

}

int main() {
   createList();
  if(Palindrome()){
      cout<<"The list is palindrome"<<endl;
  }
  else{
      cout<<"The list is not palindrome"<<endl;
  }
}
```



Q5

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

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

Node* head = NULL;
Node* tail = NULL;
Node* temp = NULL;
int n = 0;

void createList() {
   cout << "Enter the number of nodes you want: ";
   cin >> n;

   if (n < 1) {
      cout << "Enter a valid value" << endl;
      return;
   }

   cout << "Enter the values of your nodes:" << endl;
   for (int i = 0; i < n; i++) {
      Node* newNode = new Node();
      cin >> newNode->val;
```

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

bool isCircular(){
    if(tail->next==head){
        return true;
    }
    return false;
}

int main() {
    createList();
if(isCircular()){
    cout<<"The list is circular"<<endl;
}
else{
    cout<<"The list is not circular"<<endl;
}
    return 0;
```

}

Output

```
Enter the number of nodes you want: 3
Enter the values of your nodes:
1
2
3
The list is circular
```