In the Name of Allah, the Most Gracious, the Most Merciful

COMSATS UNIVERSITY ISLAMABAD

Department: Computer Science

Submitted By:

ZAIN SALEEM

Submitted To:

MAM YASMEEN JANA

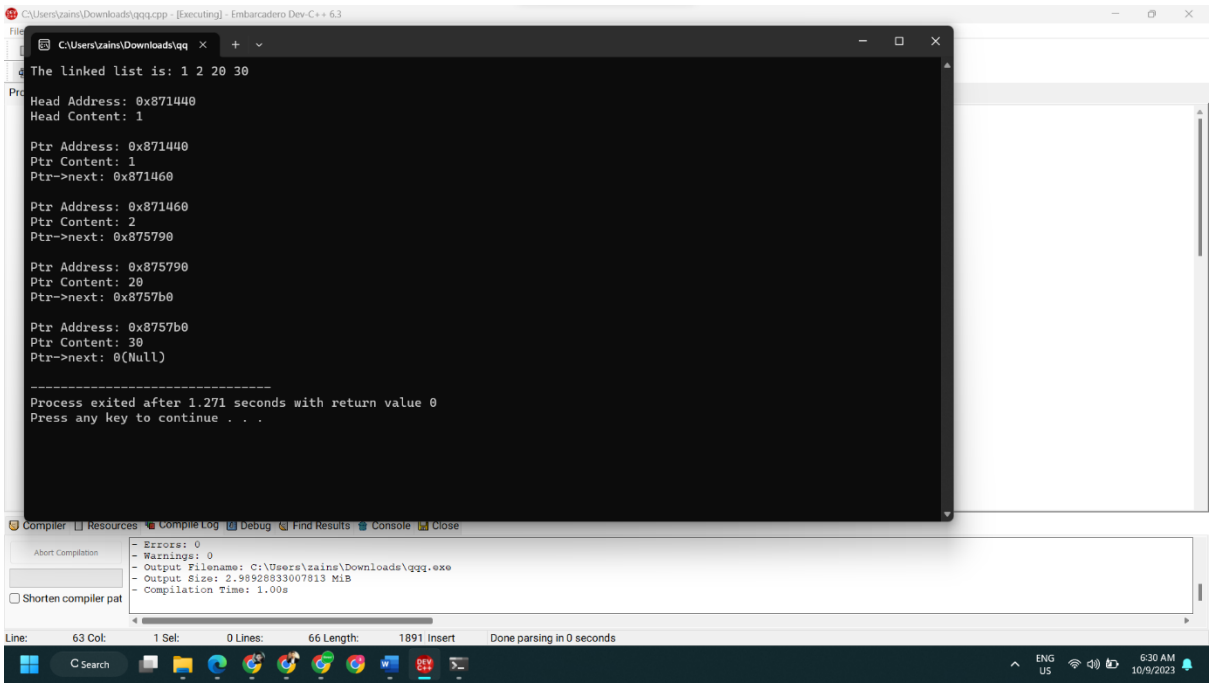
- **Course Title :** DATA STRUCTURES & ALGORITHMS (LAB)
- **Assignment NO :** 2
- **Registration No :** SP22-BCS-126
- **Section :** B
- **Date of submission :** 09/10/2023

PROGRAM # 01

```
#include <iostream>

// Define a struct for the linked list node
struct Node {
    int data;
    Node* next;
};

// Function to display the linked list and other information
void displayLinkedList(Node* head) {
    // Print the linked list content at the start
    std::cout << "The linked list is: ";
    Node* current = head;
    while (current != nullptr) {
        std::cout << current->data << " ";
        current = current->next;
    }
    std::cout << std::endl;
    std::cout << std::endl;

    // Print the head address and content
    std::cout << "Head Address: " << head << std::endl ;
    std::cout << "Head Content: " << head->data << std::endl << std::endl;

    // Traverse and print each node in the linked list
    current = head;
```

```
    while (current != nullptr) {
        // Print the current node's address and content
        std::cout << "Ptr Address: " << current << std::endl;
        std::cout << "Ptr Content: " << current->data << std::endl;

        // Print ptr->next if it exists
        if (current->next != nullptr) {
            std::cout << "Ptr->next: " << current->next << std::endl << std::endl;

        }

        // Move to the next node
        current = current->next;

    }
    std::cout << "Ptr->next: 0(Null)";
    std::cout << std::endl;

}

int main() {
    // Create a sample linked list with some nodes
    Node* head = new Node{1, nullptr};
    head->next = new Node{2, nullptr};
    head->next->next = new Node{20, nullptr};
    head->next->next->next = new Node{30, nullptr};
```

```
    // Display the linked list and related information
    displayLinkedList(head);

    // Don't forget to free the memory allocated for the nodes
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }

    return 0;
}
```



PROGRAM # 02

```
#include <iostream>

// Define a struct for a linked list node
struct Node {
    int data;
    Node* next;
    Node* prev; // For doubly linked list
};

class LinkedList {
private:
    Node* head; // Head of the linked list
    Node* tail; // Tail of the doubly linked list
    bool isCircular;

public:
    LinkedList(bool circular = false) : head(nullptr), tail(nullptr), isCircular(circular) {}

    // Function to insert a node at the beginning
    void insertAtBeginning(int value) {
        Node* newNode = new Node{value, nullptr, nullptr};
        if (!head) {
            head = newNode;
            tail = newNode;
            if (isCircular) tail->next = head;
        } else {
```

```
        newNode->next = head;
        if (isCircular) tail->next = newNode;
        if (head) head->prev = newNode; // Fix: Check if head is not null
        head = newNode;
    }
}

// Function to insert a node at the end
void insertAtEnd(int value) {
    Node* newNode = new Node{value, nullptr, nullptr};
    if (!tail) {
        head = newNode;
        tail = newNode;
        if (isCircular) tail->next = head;
    } else {
        newNode->prev = tail;
        tail->next = newNode;
        tail = newNode;
        if (isCircular) tail->next = head;
    }
}

// Function to display the linked list
void display() {
    Node* current = head;
    if (isCircular) {
        do {
```

```
            std::cout << current->data << " ";
            current = current->next;
        } while (current != head);
    } else {
        while (current != nullptr) {
            std::cout << current->data << " ";
            current = current->next;
        }
    }
    std::cout << std::endl;
}

// Function to reverse the linked list
void reverse() {
    Node* current = head;
    Node* prev = nullptr;
    Node* next = nullptr;

    while (current != nullptr) {
        next = current->next;
        current->next = prev;
        current->prev = next; // For doubly linked list
        prev = current;
        current = next;
    }

    // Update the head and tail pointers
```

```
        if (isCircular) {
            tail = head;
            head = prev;
        } else {
            head = prev;
            if (tail) tail->next = nullptr; // Fix: Update tail's next pointer to nullptr
        }
    }

    // Function to delete a node with a given value
    void deleteNode(int value) {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == value) {
                // Handle deletion based on the linked list type (single or double)
                if (current->prev != nullptr) {
                    current->prev->next = current->next;
                } else {
                    head = current->next;
                }

                if (current->next != nullptr) {
                    current->next->prev = current->prev;
                } else {
                    tail = current->prev;
                }
```

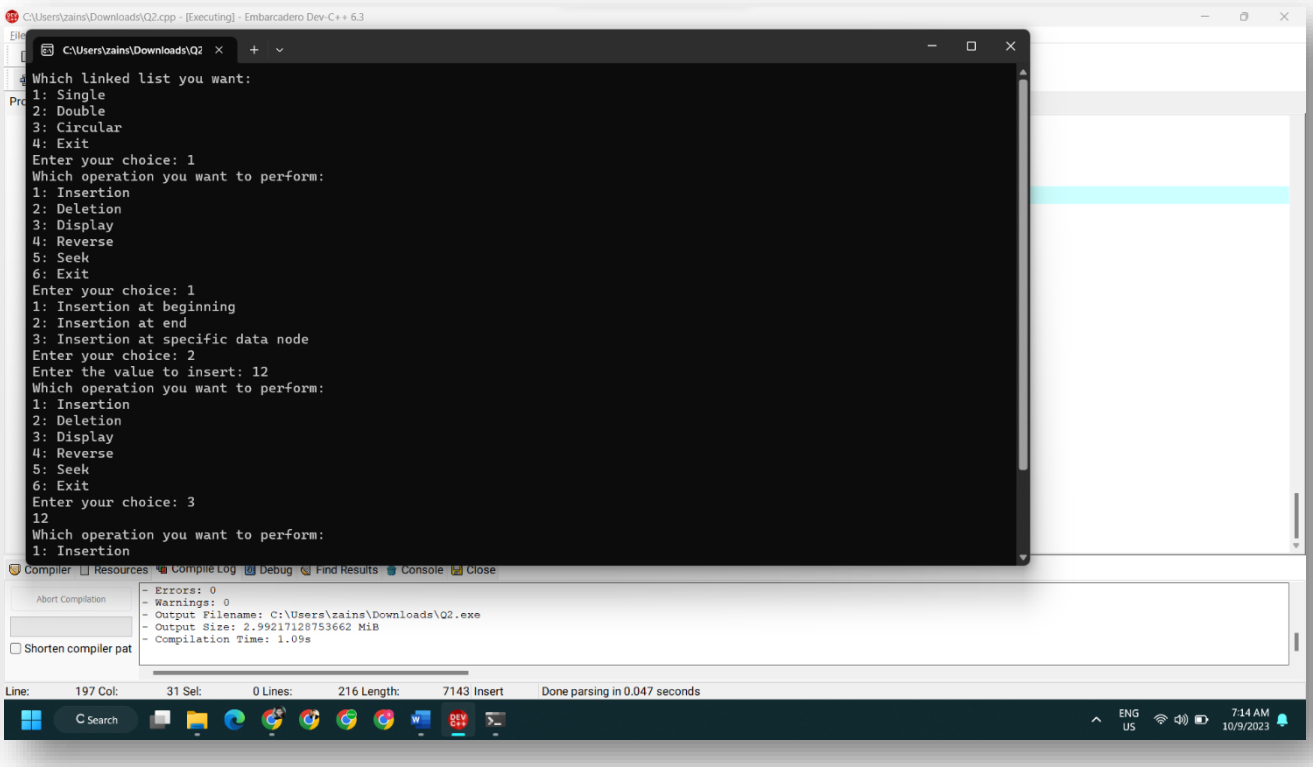
```
                delete current;
                return; // Exit after the first occurrence is deleted
            }
            current = current->next;
        }
    }
};

int main() {
    bool isRunning = true;
    int choice;
    LinkedList* list = nullptr;

    while (isRunning) {
        std::cout << "Which linked list you want:\n"
                  "1: Single\n"
                  "2: Double\n"
                  "3: Circular\n"
                  "4: Exit\n"
                  "Enter your choice: ";
        std::cin >> choice;

        switch (choice) {
            case 1:
                list = new LinkedList(false);
                break;
```

```
            case 2:
                list = new LinkedList(false);
                break;
            case 3:
                list = new LinkedList(true);
                break;
            case 4:
                isRunning = false;
                delete list; // Clean up the linked list object before exiting
                break;
            default:
                std::cout << "Invalid choice. Please try again.\n";
                continue;
        }

        if (list != nullptr) {
            while (true) {
                std::cout << "Which operation you want to perform:\n"
                          "1: Insertion\n"
                          "2: Deletion\n"
                          "3: Display\n"
                          "4: Reverse\n"
                          "5: Seek\n" // Implement this operation or remove from
menu
                          "6: Exit\n"
                          "Enter your choice: ";
                std::cin >> choice;
```

```
switch (choice) {
    case 1: {
        int insertionChoice;
        std::cout << "1: Insertion at beginning\n"
                  "2: Insertion at end\n"
                  "3: Insertion at specific data node\n"
                  "Enter your choice: ";
        std::cin >> insertionChoice;

        int value;
        std::cout << "Enter the value to insert: ";
        std::cin >> value;

        // Perform the insertion based on user's choice
        if (insertionChoice == 1) {
            list->insertAtBeginning(value);
        } else if (insertionChoice == 2) {
            list->insertAtEnd(value);
        } else if (insertionChoice == 3) {
            // Handle insertion at a specific data node
        }
        break;
    }
    case 2: {
        int value;
        std::cout << "Enter the value to delete: ";
        std::cin >> value;
```

```
                list->deleteNode(value);
                break;
            }
            case 3:
                list->display();
                break;
            case 4:
                list->reverse();
                break;
            case 5:
                // Implement seeking operation or remove from menu
                break;
            case 6:
                delete list; // Clean up the linked list object before exiting
                list = nullptr;
                break;
            default:
                std::cout << "Invalid choice. Please try again.\n";
                continue;
        }
        if (choice == 6 || list == nullptr) break;
    }
  }
}

return 0;
```

```
}
```


C:\Users\zains\Downloads\Q2.cpp - [Executing] - Embarcadero Dev-C++ 6.3
C:\Users\zains\Downloads\Q2
Which linked list you want:
1: Single
2: Double
3: Circular
4: Exit
Enter your choice: 1
Which operation you want to perform:
1: Insertion
2: Deletion
3: Display
4: Reverse
5: Seek
6: Exit
Enter your choice: 1
1: Insertion at beginning
2: Insertion at end
3: Insertion at specific data node
Enter your choice: 2
Enter the value to insert: 12
Which operation you want to perform:
1: Insertion
2: Deletion
3: Display
4: Reverse
5: Seek
6: Exit
Enter your choice: 3
12
Which operation you want to perform:
1: Insertion
Compiler Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\zains\Downloads\Q2.exe
- Output Size: 2.9921712875366 2 MiB
- Compilation Time: 1.09s
Line: 197 Col: 31 Sel: 0 Lines: 216 Length: 7143 Insert
Done parsing in 0.047 seconds
Search
ENG
US
7:14 AM
10/9/2023