REVERSRING, SEARCHING AND MERGING A SINGLY LINKED LIST

CODE:

```
#include <iostream>
class Node {
public:
    int data;
    Node* next;
    Node(int data) : data(data), next(nullptr) {}
};
class LinkedList {
public:
    Node* head;
    LinkedList() : head(nullptr) {}
    // Add a node at the end of the list
    void append(int data) {
        Node* newNode = new Node(data);
        if (!head) {
            head = newNode;
            }
        else {
            Node* current = head;
            while (current->next) {
            current = current->next;
            }
            current->next = newNode;
        }
```

```
    }
// Reverse the linked list
    void reverse() {
    Node* prev = nullptr;
    Node* current = head;
    Node* next = nullptr;
    while (current != nullptr) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head = prev;
}
// Search for a value in the linked list
bool search(int value) {
    Node* current = head;
        while (current) {
            if (current->data == value) {
            return true;
        }
        current = current->next;
    }
    return false;
    }
// Merge two linked lists
void merge(LinkedList& otherList) {
    if (!otherList.head) {
```

```
        return;
    }
    if (!head) {
        head = otherList.head;
        return;
    }
    Node* current = head;
    while (current->next) {
        current = current->next;
    }
    current->next = otherList.head;
    otherList.head = nullptr;
}
// Display the linked list
void display() {
    Node* current = head;
    while (current) {
        std::cout << current->data << " -> ";
        current = current->next;
    }
    std::cout << "nullptr" << std::endl;
    }
};
int main() {
    LinkedList list;
    list.append(1);
    list.append(2);
    list.append(3);
```

```
    list.append(4);
    std::cout << "Original Linked List: ";
    list.display();
    list.reverse();
    std::cout << "Reversed Linked List: ";
    list.display();
    int searchValue = 7;
    if (list.search(searchValue)) {
        std::cout << searchValue << ": FOUND IN THE
LIST!" << std::endl;
    }
    else {
        std::cout << searchValue << ": NOT FOUND IN
THE LIST!" << std::endl;
    }
    LinkedList otherList;
    otherList.append(5);
    otherList.append(6);
    list.merge(otherList);
    std::cout << "Merged Linked List: ";
    list.display();
    return 0;
}
```

OUTPUT:

```
Original Linked List: 1 -> 2 -> 3 -> 4 -> nullptr
Reversed Linked List: 4 -> 3 -> 2 -> 1 -> nullptr
3: FOUND IN THE LIST!
Merged Linked List: 4 -> 3 -> 2 -> 1 -> 5 -> 6 -> nullptr
```

```
Original Linked List: 1 -> 2 -> 3 -> 4 -> nullptr
Reversed Linked List: 4 -> 3 -> 2 -> 1 -> nullptr
7: NOT FOUND IN THE LIST!
Merged Linked List: 4 -> 3 -> 2 -> 1 -> 5 -> 6 -> nullptr
```