Assignment: Implement a Single Linked List in C++

Objective: Develop a single linked list in C++ to practice dynamic memory management, class design, and fundamental operations.

Problem Statement

You are required to create a Single Linked List (SLL) implementation in C++ that supports the following operations:

1. Insert:
   - Insert a node at the beginning.
   - Insert a node at the end.
   - Insert a node after a given value.
2. Delete:
   - Delete a node by its value.
   - Delete the first node.
   - Delete the last node.
3. Search:
   - Search for a node by its value and return its position.
4. Display:
   - Display all elements of the linked list in sequential order.
5. Reverse:
   - Reverse the linked list.

Instructions

1. Class Design:

- Create a class `Node` with the following members:

  - `int data` (to store the value).

  - `Node* next` (to store the address of the next node).

- Create another class `LinkedList` to implement all the required functionalities.

2. Implementation Guidelines:

- Use dynamic memory allocation to create and delete nodes.

- Handle edge cases such as inserting into an empty list or deleting from an empty list.

- Ensure no memory leaks by freeing memory when nodes are deleted.

3. Functions to Implement:

- `void insertAtBeginning(int value)`

- `void insertAtEnd(int value)`

- `void insertAfter(int searchValue, int newValue)`

- `void deleteByValue(int value)`

- `void deleteFirst()`

- `void deleteLast()`

- `int search(int value)`

- `void display()`

- `void reverse()`

4. Main Program:

- Create an instance of the `LinkedList` class in the `main` function.

- Demonstrate all the implemented functionalities with user inputs (interactive menu-driven program).

Evaluation Criteria

1. Correct implementation of all operations.

2. Proper handling of edge cases.

3. Clear and structured code with comments for each function.

4. No memory leaks (ensure all dynamically allocated nodes are freed).

Deliverables: Submit the C++ source code file with all functionalities implemented along with screenshot of output in blackboard and cpp file to git.

Sample output:

```
Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 8
Current Linked List: List is empty.

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 1
Enter the value to insert at the beginning: 19
```

```
Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 2
Enter the value to insert at the end: 22

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 8
Current Linked List: 19 -> 22 -> NULL
```

```
Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 1
Enter the value to insert at the beginning: 20

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 8
Current Linked List: 20 -> 19 -> 22 -> NULL
```

```
Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 2
Enter the value to insert at the end: 15

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 8
Current Linked List: 20 -> 19 -> 22 -> 15 -> NULL
```

```
Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 7
Enter the value to search: 35
Value not found in the list.

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 9
Reversed Linked List: 15 -> 22 -> 19 -> 20 -> NULL
```

```
Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 8
Current Linked List: 15 -> 22 -> 19 -> 20 -> NULL

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 9
Reversed Linked List: 20 -> 19 -> 22 -> 15 -> NULL

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
```

```
Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 9
Reversed Linked List: 20 -> 19 -> 22 -> 15 -> NULL

Single Linked List Operations:
1. Insert at Beginning
2. Insert at End
3. Insert After Value
4. Delete by Value
5. Delete First Node
6. Delete Last Node
7. Search for Value
8. Display List
9. Reverse List
10. Exit
Enter your choice: 10
Exiting program.
```