Bahria University, Islamabad

Department of Software Engineering

Data Structre And Algorithms

(Fall-2024)

Teacher: Engr. Aleem Ahmad

Student : Lotfullah Muslimwal

Enrollment : 01-131232-039

Lab Journal: X

Date:

Task No:	Task Wise Marks		Documentation Marks		Total Marks (20)
	Assigned	Obtained	Assigned	Obtained	
1	3		5		
2	3				
3	3				
4	3				
5	3				

Comments:

Signature

LAB

Lab No: 06

Task 1: Singly Linked List with Insertions, Deletions, and Display Operations

Code:

```
#include <iostream>
#include <limits>
#include <sstream>  // For stringstream
using namespace std;

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

// Insert at the beginning
void insertAtStart(Node*& head, int value) {
   Node* newNode = new Node();
   newNode->data = value;
   newNode->next = head;
   head = newNode;
   cout << value << " inserted at the start." << endl;
}

// Insert at the end
void insertAtEnd(Node*& head, int value) {
   Node* newNode = new Node();
   newNode->data = value;
   newNode->next = nullptr;
   if (head == nullptr) {
      head = newNode;
   }
   else {
      Node* temp = head;
      while (temp->next != nullptr) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
   cout << value << " inserted at the end." << endl;
}
```

```
// Insert after a specific node
void insertAfter(Node* prevNode, int value) {
    if (prevNode == nullptr) {
        cout << "The given previous node cannot be NULL." << endl;
        return;
    }
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = prevNode->next;
    prevNode->next = newNode;
    cout << value << " inserted after node with value " << prevNode->data << "." << endl;
}

// Delete at the end
void deleteAtEnd(Node*& head) {
    if (head == nullptr) {
        cout << "List is empty. Nothing to delete." << endl;
        return;
    }
    if (head->next == nullptr) {
        delete head;
        head = nullptr;
        cout << "Last node deleted." << endl;
        return;
    }
    Node* temp = head;
    while (temp->next->next != nullptr) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = nullptr;
    cout << "Last node deleted." << endl;
}

// Delete after a specific node
void deleteAfter(Node* prevNode) {
    if (prevNode == nullptr || prevNode->next == nullptr) {
        cout << "No node to delete after the given node." << endl;
        return;
    }
    Node* temp = prevNode->next;
    prevNode->next = temp->next;
    delete temp;
    cout << "Node deleted after node with value " << prevNode->data << "." << endl;
}

// Display the list
void displayList(Node* head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}

// Input validation function to get a valid integer choice for the menu
// Input validation function to get a valid integer choice for the menu
int getValidatedChoice(int min, int max) {
    string input;
    int choice;

    while (true) {
        getline(cin, input);  // Read the entire line as a string

        // Check if input is empty
        if (input.empty()) {
            cout << "Empty input. Please enter a number between " << min << " and " << max <<
": ";
            continue;
        }

        // Try to convert string input to an integer
        stringstream ss(input);
        if (ss >> choice && choice >= min && choice <= max) {
            return choice;  // Return the valid choice within the specified range
        }
        else {
            cout << "Invalid choice. Please enter a number between " << min << " and " << max
<< ": ";
        }
    }
}
```

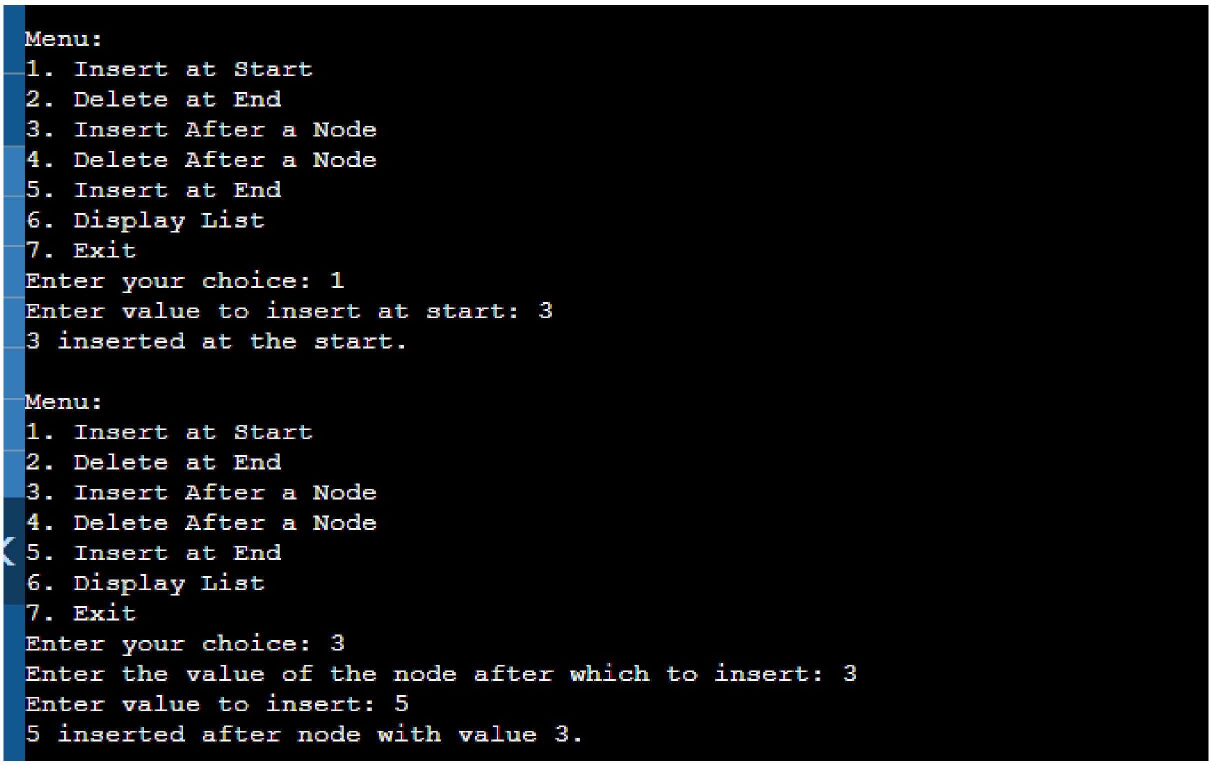
```
// Input validation function to get a valid integer for node values
int getValidInteger(const string& prompt) {
   string input;
   int value;

   while (true) {
      cout << prompt;
      getline(cin, input);  // Read the entire line as a string

      // Check if input is empty
      if (input.empty()) {
         cout << "Empty input. Please enter a valid integer." << endl;
         continue;
      }

      // Try to convert string input to an integer
      stringstream ss(input);
      if (ss >> value) {
         return value;  // Return the valid integer
      }
      else {
         cout << "Invalid input. Please enter a valid integer." << endl;
      }
   }
}


int main() {
   Node* head = nullptr;
   int choice, value, afterValue;

   do {
      cout << "\nMenu:\n";
      cout << "1. Insert at Start\n";
      cout << "2. Delete at End\n";
      cout << "3. Insert After a Node\n";
      cout << "4. Delete After a Node\n";
      cout << "5. Insert at End\n";
      cout << "6. Display List\n";
      cout << "7. Exit\n";
      cout << "Enter your choice: ";

      choice = getValidatedChoice(1, 7);  // Validate menu choice input
```

```
    switch (choice) {
    case 1:
        value = getValidInteger("Enter value to insert at start: ");  // Validate node value input
        insertAtStart(head, value);
        break;

    case 2:
        deleteAtEnd(head);
        break;

    case 3:
        afterValue = getValidInteger("Enter the value of the node after which to insert: ");  //
Validate node position value
        value = getValidInteger("Enter value to insert: ");  // Validate new node value
        {
            Node* temp = head;
            while (temp != nullptr && temp->data != afterValue) {
                temp = temp->next;
            }
            if (temp != nullptr) {
                insertAfter(temp, value);
            }
            else {
                cout << "Node with value " << afterValue << " not found." << endl;
            }
        }
        break;

    case 4:
        afterValue = getValidInteger("Enter the value of the node after which to delete: ");  //
Validate node position for deletion
        {
            Node* temp = head;
            while (temp != nullptr && temp->data != afterValue) {
                temp = temp->next;
            }
            if (temp != nullptr) {
                deleteAfter(temp);
            }
            else {
                cout << "Node with value " << afterValue << " not found." << endl;
            }
        }
```

```
            break;

        case 5:
            value = getValidInteger("Enter value to insert at end: ");  // Validate node value for
end insertion
            insertAtEnd(head, value);
            break;

        case 6:
            cout << "Current List: ";
            displayList(head);
            break;

        case 7:
            cout << "Exiting..." << endl;
            break;

        default:
            cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 7);

    return 0;
}
```

GitHub-Link: https://github.com/lotfullahmsl/DSA-Lab-FA2024

Screenshot:



```
Menu:
1. Insert at Start
2. Delete at End
3. Insert After a Node
4. Delete After a Node
5. Insert at End
6. Display List
7. Exit
Enter your choice: 1
Enter value to insert at start: 3
3 inserted at the start.

Menu:
1. Insert at Start
2. Delete at End
3. Insert After a Node
4. Delete After a Node
5. Insert at End
6. Display List
7. Exit
Enter your choice: 3
Enter the value of the node after which to insert: 3
Enter value to insert: 5
5 inserted after node with value 3.
```