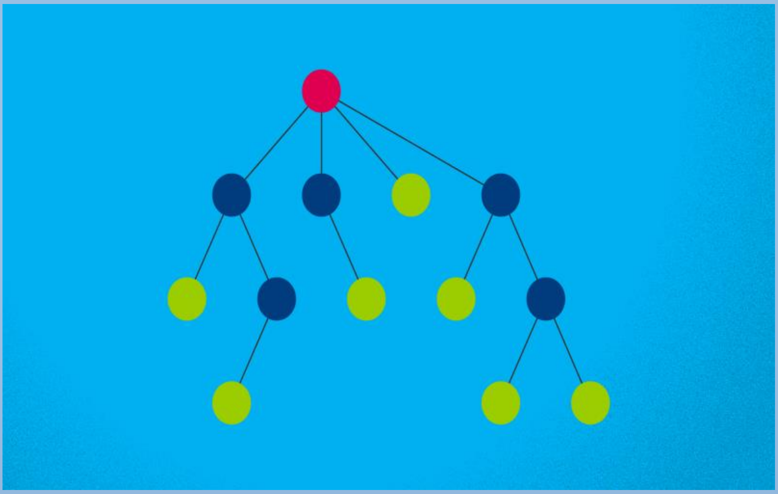
RIPHAH INTERNATIONAL UNIVERSITY

DATA STRUCTURE & ALGORITHMS



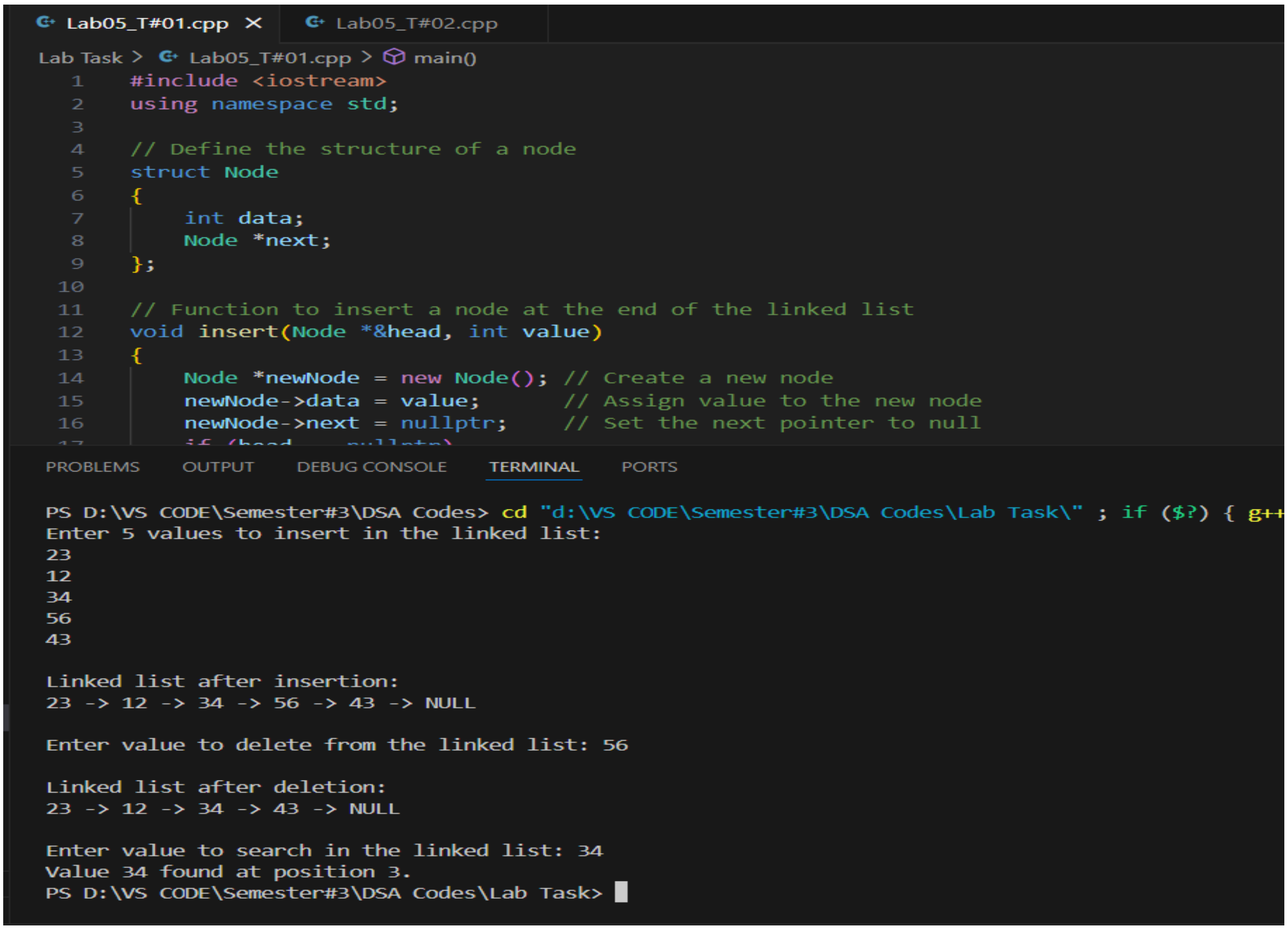
LAB # 05

NAME	: ASAD ULLAH
SAP ID	: 55181
SECTION	: SE 3-2

CODE # T-01 :

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

// Define the structure of a node
struct Node
{
    int data;
    Node *next;
};

// Function to insert a node at the end of the linked list
void insert(Node *&head, int value)
{
    Node *newNode = new Node(); // Create a new node
    newNode->data = value;      // Assign value to the new node
    newNode->next = nullptr;    // Set the next pointer to null
    if (head == nullptr)
    {
        head = newNode;         // If the list is empty, make this node the head
    }
    else
    {
        Node *temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next; // Traverse to the end of the list
        }
        temp->next = newNode; // Add the new node at the end of the list
    }
}

// Function to delete a node by value
void deleteNode(Node *&head, int value)
{
    if (head == nullptr)
    {
        cout << "List is empty. Cannot delete.\n";
        return;
    }

    if (head->data == value)
    { // If the node to be deleted is the head node
        Node *temp = head;
        head = head->next; // Update the head pointer
        delete temp;
        return;
    }
    Node *current = head;
    Node *prev = nullptr;
    while (current != nullptr && current->data != value)
    {
        prev = current;
        current = current->next;
    }
    if (current == nullptr)
    {
        cout << "Node with value " << value << " not found.\n";
        return;
    }

    prev->next = current->next; // Remove the node from the list
    delete current;
}

// Function to search for a node by value
void search(Node *head, int value)
{
    Node *temp = head;
    int pos = 1;
    while (temp != nullptr)
    {
        if (temp->data == value)
        {
            cout << "Value " << value << " found at position " << pos << ".\n";
            return;
        }
        temp = temp->next;
        pos++;
    }
    cout << "Value " << value << " not found in the list.\n";
}

// Function to display the linked list
void display(Node *head)
{
    if (head == nullptr)
    {
        cout << "List is empty.\n";
        return;
    }

    Node *temp = head;
    while (temp != nullptr)
    {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

int main()
{
    Node *head = nullptr;
    int value;
    // Taking input for 5 nodes
    cout << "Enter 5 values to insert in the linked list:"<<endl;
    for (int i = 0; i < 5; i++)
    {
        cin >> value;
        insert(head, value);
    }

    cout << "\nLinked list after insertion: "<<endl;
    display(head);

    // Deletion
    cout << "\nEnter value to delete from the linked list: ";
    cin >> value;
    deleteNode(head, value);
    cout << "\nLinked list after deletion: "<<endl;
    display(head);

    // Search
    cout << "\nEnter value to search in the linked list: ";
    cin >> value;
    search(head, value);

    return 0;
}
```

OUTPUT # T-01 :



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

// Define the structure of a node
struct Node
{
    int data;
    Node *next;
};

// Function to insert a node at the end of the linked list
void insert(Node *&head, int value)
{
    Node *newNode = new Node(); // Create a new node
    newNode->data = value;      // Assign value to the new node
    newNode->next = nullptr;    // Set the next pointer to null
```

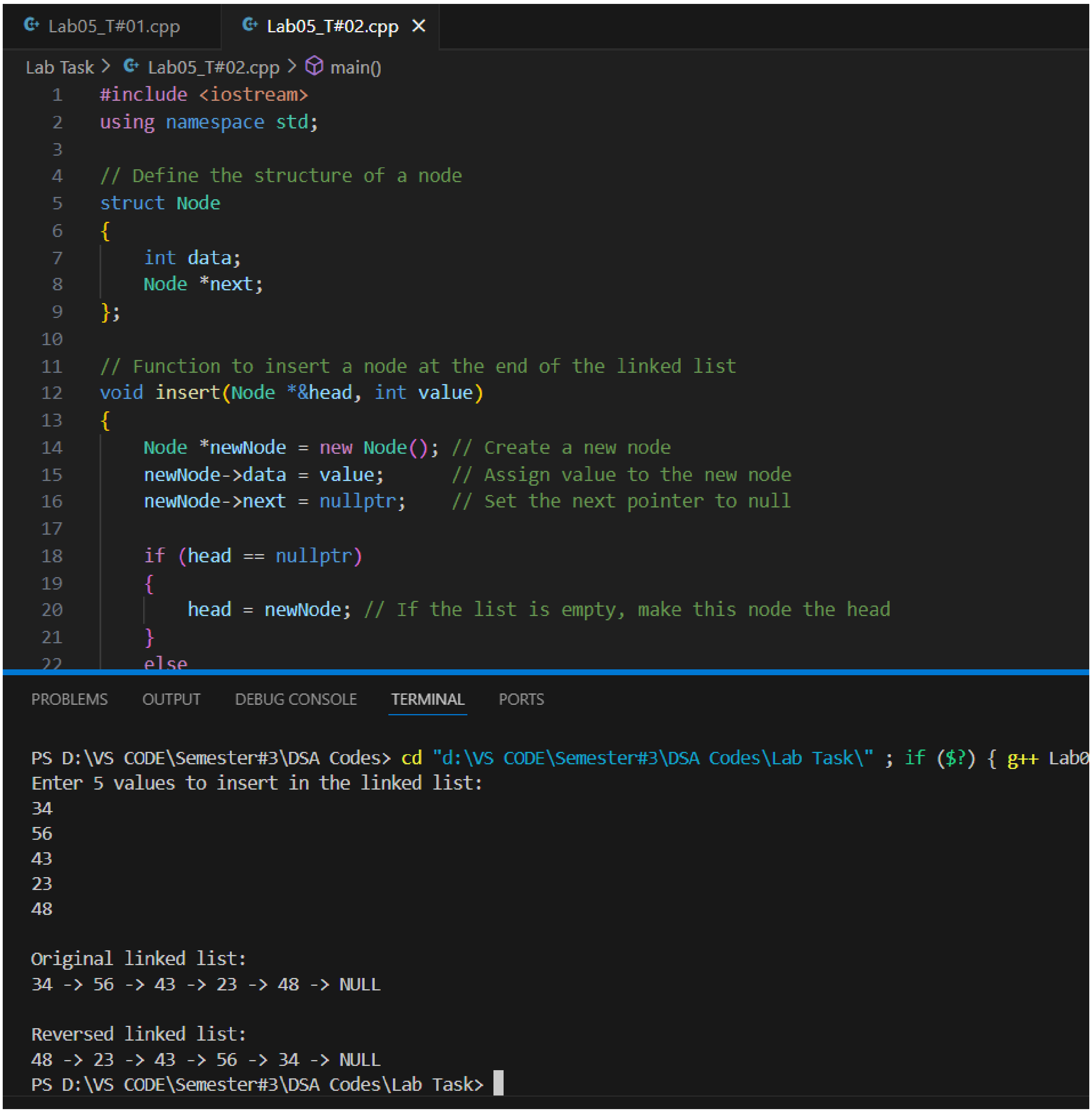
```
PS D:\VS CODE\Semester#3\DSA Codes> cd "d:\VS CODE\Semester#3\DSA Codes\Lab Task\" ; if ($?) { g++
Enter 5 values to insert in the linked list:
23
12
34
56
43

Linked list after insertion:
23 -> 12 -> 34 -> 56 -> 43 -> NULL

Enter value to delete from the linked list: 56

Linked list after deletion:
23 -> 12 -> 34 -> 43 -> NULL

Enter value to search in the linked list: 34
Value 34 found at position 3.
PS D:\VS CODE\Semester#3\DSA Codes\Lab Task>
```

CODE # T-02 :

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

// Define the structure of a node
struct Node
{
    int data;
    Node *next;
};

// Function to insert a node at the end of the linked list
void insert(Node *&head, int value)
{
    Node *newNode = new Node(); // Create a new node
    newNode->data = value;      // Assign value to the new node
    newNode->next = nullptr;    // Set the next pointer to null

    if (head == nullptr)
    {
        head = newNode; // If the list is empty, make this node the head
    }
    else
    {
        Node *temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next; // Traverse to the end of the list
        }
        temp->next = newNode; // Add the new node at the end of the list
    }
}

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

    while (current != nullptr)
    {
        next = current->next; // Store the next node
        current->next = prev; // Reverse the current node's pointer
        prev = current;       // Move the `prev` to current
        current = next;       // Move to the next node
    }
    head = prev; // Update the head to the new first node
}

// Function to display the linked list
void display(Node *head)
```

```cpp
{
    if (head == nullptr)
    {
        cout << "List is empty.\n";
        return;
    }

    Node *temp = head;
    while (temp != nullptr)
    {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

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

    // Insert some values into the linked list
    cout << "Enter 5 values to insert in the linked list:"<<endl;
    for (int i = 0; i < 5; i++)
    {
        cin >> value;
        insert(head, value);
    }

    cout << "\nOriginal linked list: "<<endl;
    display(head);

    // Reverse the linked list
    reverse(head);
    cout << "\nReversed linked list: "<<endl;
    display(head);

    return 0;
}
```

OUTPUT # T-02 :



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

// Define the structure of a node
struct Node
{
    int data;
    Node *next;
};

// Function to insert a node at the end of the linked list
void insert(Node *&head, int value)
{
    Node *newNode = new Node(); // Create a new node
    newNode->data = value;      // Assign value to the new node
    newNode->next = nullptr;    // Set the next pointer to null

    if (head == nullptr)
    {
        head = newNode; // If the list is empty, make this node the head
    }
    else
```

```
PS D:\VS CODE\Semester#3\DSA Codes> cd "d:\VS CODE\Semester#3\DSA Codes\Lab Task\" ; if ($?) { g++ Lab0
Enter 5 values to insert in the linked list:
34
56
43
23
48

Original linked list:
34 -> 56 -> 43 -> 23 -> 48 -> NULL

Reversed linked list:
48 -> 23 -> 43 -> 56 -> 34 -> NULL
PS D:\VS CODE\Semester#3\DSA Codes\Lab Task>
```