Group 6

DEVOPS Assignment

Our Assignment outlines the collaborative development and integration process followed by eight contributors. The assignment involves the implementation of coding functionalities, continuous integration, and code testing for which the respective images for Code Input, Code Output, Team Members, and Commit History are provided below.

Our Team Members:

21bds014 - Chinmay Haval
21bds025 - Kartik Jagtap
21bds037 - Mayank Maurya
21bds038 - Milind Murmu
21bds044 - Nikhil Baghel
21bds054 - Rahul Singh
21bds058 - Saksham Sharma
21bds062 - Pranjal Shinde(**Admin**)

Primary Objective :

The primary objective of the Assignment is to create a proper DevOps Pipeline at the Code Integration and the Code testing stage where the Admin creates a new Repository and adds a file (.cpp file) in our case and The Admin invites Collaborators to join and fork the repository so that they can pull the recent changes from the repository and add their specific Functions in the .cpp file individually respectively and finally, after adding, Committing(with proper message), and Pushing the code (Functions) , the Collaborators push the changes in their forked repository and create,

review properly and then send a pull request to the Admin which can then be merged properly into the Admin repository.

Here, we also experience **certain difficulties** .

In our project, with eight members working on different functionalities within the same codebase, merge conflicts were a significant challenge. These conflicts occurred when multiple contributors modified the same files or lines of code, and Git couldn't automatically reconcile the changes. Shared resources like configuration files and core logic were particularly prone to this. Merge conflicts typically arose because several team members edited the same code simultaneously, made changes without knowing about others' contributions, or restructured files. These conflicts could have delayed our project without a solid strategy and disrupted the workflow.

Our Project Home Page :
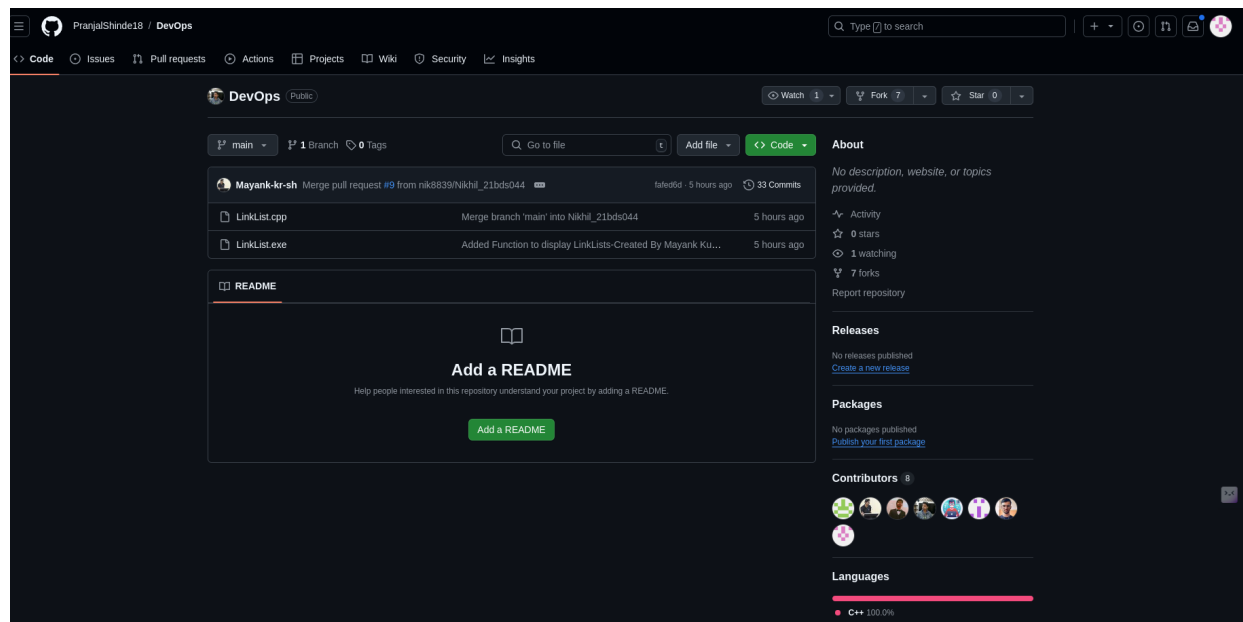
Code and Functions that we have Used:
1. Add to end of list:

```cpp
// Mayank Kumar Shah(21bds037)
void addToEnd(int value){
    // Create a new node
    Node* newNode = new Node(value);
    // If the list is empty, make the new node the head
    if (head == nullptr) {
        head = newNode;
        return;
    }
    // Otherwise, traverse to the last node
    Node* temp = head;
    while (temp->next != nullptr) {
        temp = temp->next;
    }
    // Add the new node at the end
    temp->next = newNode;
}
```

2. Add node to the Middle of the Linked List:

```cpp
// Function to add a node in the middle of the linked list
// Chinmay (21bds014)
void addToMiddle(int value, int position)
{

    Node *node = new Node(value);
    Node *prev = nullptr;
    Node *curr = head;
    if (position == 0)
    {
        node->next = head;
        head = node;
    }
    while (node != nullptr && position != 0)
    {
        prev = curr;
        curr = curr->next;
        position--;
    }
    prev->next = node;
    node->next = curr;
    return;
}
```

3. Delete node from the end of the Linked list:

```cpp
// Function to delete the last node of the linked list
// Milind (21bds038)
void deleteFromEnd()
{
    if (head == nullptr) { // If the list is empty
        cout << "The list is empty, nothing to delete.\n";
        return;
    }

    if (head->next == nullptr) { // If there is only one node
        delete head; // Delete the head
        head = nullptr; // Set head to null
        cout << "Last node deleted. The list is now empty.\n";
        return;
    }

    Node* current = head;
    while (current->next->next != nullptr) {
        current = current->next;
    }

    // Delete the last node
    delete current->next;
    current->next = nullptr; // Set the second last node's next to
    cout << "Last node deleted.\n";
}
```

4. Delete from the Middle of the Linked List:

```cpp
// Function to delete a node from the middle of the linked list
// Rahul Singh (21bds054)
void deleteFromMiddle(int position) {
    if (head == nullptr) {
        cout << "The list is empty, nothing to delete.\n";
        return;
    }

    Node* curr = head;
    Node* prev = nullptr;

    if (position == 0) {
        head = curr->next;
        delete curr;
        cout << "Node at position 0 deleted.\n";
        return;
    }

    for (int i = 0; curr != nullptr && i < position; i++) {
        prev = curr;
        curr = curr->next;
    }

    if (curr == nullptr) {
        cout << "Position out of bounds.\n";
        return;
    }

    prev->next = curr->next;
    delete curr;
    cout << "Node at position " << position << " deleted.\n";
}
```

5. Reverse Linked list:

```cpp
// Function to reverse the linked list
// Saksham (21bds058)
void reverse() {

    Node* current = head; // Pointer to the current node
    Node* prev = nullptr; // Pointer to the previous node
    Node* next = nullptr; // Pointer to the next node

    while (current != nullptr) {
        next = current->next; // Store the next node
        current->next = prev; // Point the current node to the previous node
        prev = current; // Move the previous node to the current node
        current = next; // Move the current node to the next node
    }

    head = prev; // Update the head of the list

    // Display the reversed linked list
    cout << "Reversed Linked List: ";
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
```

6. Deleted all even nodes :

```cpp
// Function to delete all even nodes from the linked list
// Nikhil (21bds044)
void deleteAllEvenNodes()
{
        while (head != nullptr && head->data % 2 == 0) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }


    Node* current = head;
    while (current != nullptr && current->next != nullptr) {
        if (current->next->data % 2 == 0) {

            Node* temp = current->next;
            current->next = current->next->next;
            delete temp;
        } else {

            current = current->next;
        }
    }
}
```

7. Deleted all odd nodes :

```cpp
// Function to delete all odd nodes from the linked list
// Kartik (21bds025)
void deleteAllOddNodes(){
    if (head == nullptr) {
    cout << "The list is empty.\n";
    return;
    }

    Node* temp = head;
    Node* prev = nullptr;
    int position = 1;

// If the head is at an odd position, delete it
    while (temp != nullptr && position % 2 != 0) {
        head = temp->next; // Update head to the next node
        delete temp;        // Delete the current head
        temp = head;        // Move to the next node
        position++;         // Increment position
    }

// Traverse the rest of the list and delete odd-positioned nodes
    while (temp != nullptr && temp->next != nullptr) {
        if (position % 2 != 0) {
            Node* oddNode = temp->next;
            temp->next = oddNode->next; // Bypass the odd-positioned node
            delete oddNode;             // Delete the odd-positioned node
        } else {
            temp = temp->next;          // Move to the next node
        }
        position++;                     // Increment position
    }

    cout << "All odd-positioned nodes have been deleted.\n";
}
```

8. Display

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

    Node* temp = head; // Start at the head node
    cout << "Linked List: ";
    while (temp != nullptr) { // Traverse the list
        cout << temp->data << " "; // Print each node's data
        temp = temp->next; // Move to the next node
    }
    cout << endl;
}
```

Merge Commits History:

Commit	Author	Verified	Hash
Merge pull request PranjalShinde18#2 from ksj6/main	ksj6 committed 47 minutes ago	Verified	591a959
Merge branch 'PranjalShinde18:main' into main	milind64 committed 49 minutes ago	Verified	b82c098
Milind Murmu (21BDS038) Commpleted Task: added a function to remove the last node from the LinkedList	milind64 committed 49 minutes ago		4db47e6
Linked list delete odd nodes code (Kartik Jagtap-21bds025)	ksj6 committed 50 minutes ago		e2373e8
modified the function of delete even nodes in LinkList	nik8839 committed 51 minutes ago		57cd6b0
reverse function implemented	Saksham131102 committed 1 hour ago		a784091
Merge pull request PranjalShinde18#1 from Mayank-kr-sh/main	Mayank-kr-sh committed 1 hour ago	Verified	8b950d0
Added Function to add element to end of LinkLists-Created By Mayank Kumar Shah(21BDS037)	Mayank-kr-sh committed 1 hour ago		a62c7ce
Added the driver code	Pranjal Shinde committed 1 hour ago		a62bd1a

Outputs Obtained from testing individual Functions (8) of them:

1. Add node to end of Linked List:

```
Enter your choice: 1
Enter value to add to end: 15
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 1
Enter value to add to end: 16
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 8
Linked List: 11 12 13 14 15 16
```

2. Add node to the Middle of the Linked List:

```
Enter your choice: 2
Enter value and position to add in middle: 17 2
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 8
Linked List: 11 12 17 13 14 15 16
```

3. Delete node from the end of the Linked list:

```
Enter your choice: 3
Last node deleted.
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 8
Linked List: 11 12 17 13 14 15
```

4. Delete from the Middle of the Linked List:

```
Enter your choice: 4
Enter position to delete from middle: 3
Node at position 3 deleted.
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 8
Linked List: 11 12 17 14 15
```

5. Reverse Linked list:

```
Enter your choice: 5
Reversed Linked List: 15 14 17 12 11
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 8
Linked List: 15 14 17 12 11
```

6. Deleted all even nodes :

```
Enter your choice: 6
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 8
Linked List: 15 17 11
```

7. Delete all odd nodes :

```
Enter your choice: 7
All odd-positioned nodes have been deleted.
Choose an action:
1. Add to End
2. Add to Middle
3. Delete from End
4. Delete from Middle
5. Reverse Linked List
6. Delete All Even Nodes
7. Delete All Odd Nodes
8. Display List
0. Exit
Enter your choice: 8
Linked List: 17 11
```

These are the Outputs from all the functions .