**NAME : PREM SAH**
**PRN : 123B1B234**
**DIV : D**

# ASSIGNMENT NO.4

**Implement a simple text editor application using a doubly linked list to manage the text buffer. Text editor should support the following functionalities:**
**a. Insert text**
**b. Delete text**
**c. Display text**
**d. Search text**
**e. Print text in reverse**

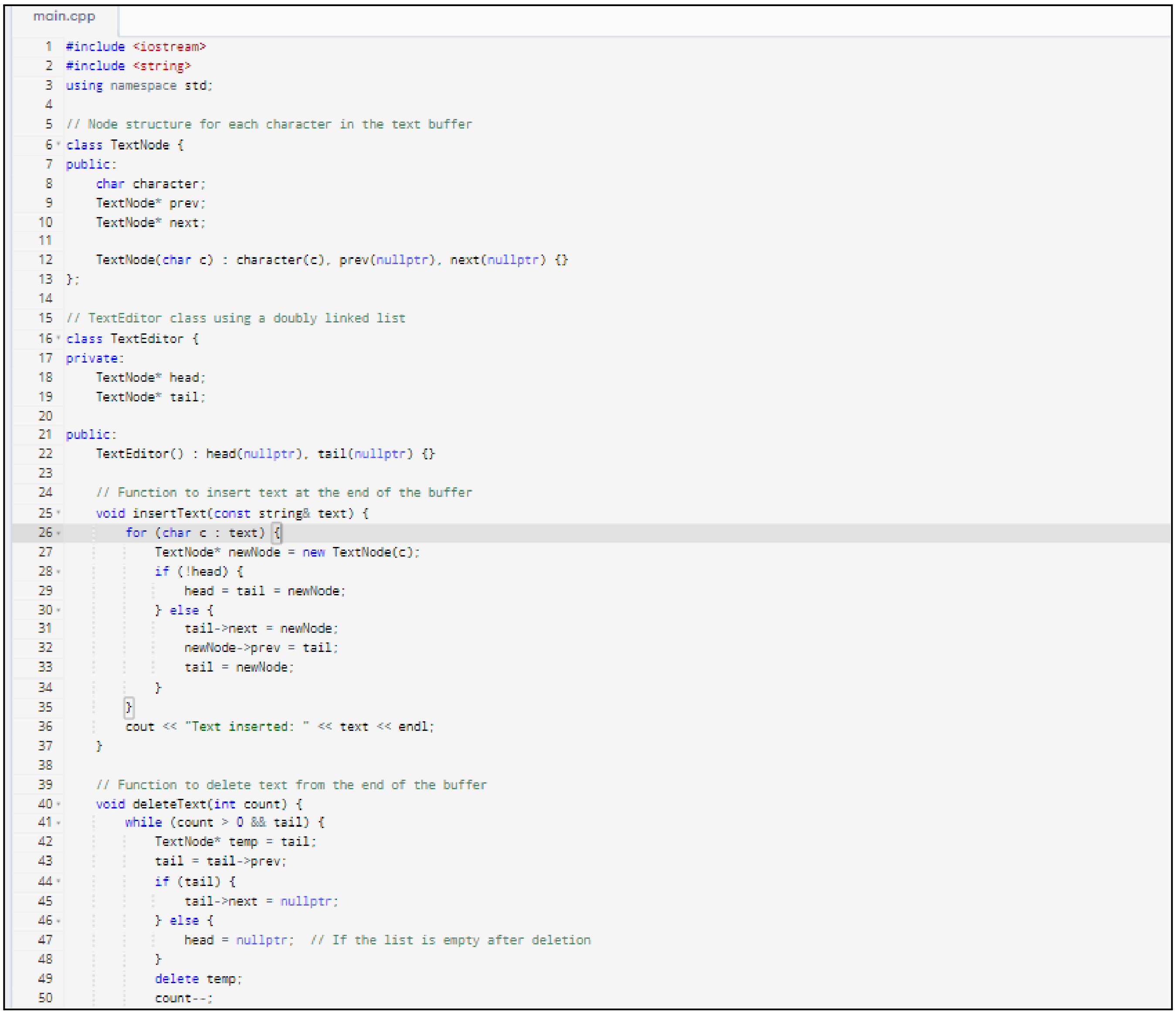
**CODE :**

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

// Node structure for each character in the text buffer
class TextNode {
public:
   char character;
   TextNode* prev;
   TextNode* next;

   TextNode(char c) : character(c), prev(nullptr), next(nullptr) {}
};

// TextEditor class using a doubly linked list
class TextEditor {
private:
   TextNode* head;
   TextNode* tail;

public:
   TextEditor() : head(nullptr), tail(nullptr) {}

   // Function to insert text at the end of the buffer
   void insertText(const string& text) {
      for (char c : text) {
         TextNode* newNode = new TextNode(c);
         if (!head) {
            head = tail = newNode;
         } else {
            tail->next = newNode;
            newNode->prev = tail;
```

```cpp
            tail = newNode;
        }
    }
    cout << "Text inserted: " << text << endl;
}

// Function to delete text from the end of the buffer
void deleteText(int count) {
    while (count > 0 && tail) {
        TextNode* temp = tail;
        tail = tail->prev;
        if (tail) {
            tail->next = nullptr;
        } else {
            head = nullptr;  // If the list is empty after deletion
        }
        delete temp;
        count--;
    }
    cout << "Deleted " << count << " characters from the end." << endl;
}

// Function to display the current text buffer
void displayText() {
    if (!head) {
        cout << "The text buffer is empty!" << endl;
        return;
    }

    TextNode* temp = head;
    cout << "Text buffer: ";
    while (temp) {
        cout << temp->character;
        temp = temp->next;
    }
    cout << endl;
}

// Function to search for a specific substring in the text buffer
void searchText(const string& query) {
    if (!head) {
        cout << "The text buffer is empty!" << endl;
        return;
    }

    TextNode* temp = head;
    bool found = false;
    int pos = 0;

    while (temp) {
            TextNode* current = temp;
            bool match = true;

            for (char c : query) {
                if (!current || current->character != c) {
                    match = false;
                    break;
                }
                current = current->next;
            }

            if (match) {
                found = true;
                cout << "Found '" << query << "' at position " << pos << endl;
                break;
            }

            temp = temp->next;
            pos++;
        }

        if (!found) {
            cout << "Text '" << query << "' not found in the buffer." << endl;
        }
    }

    // Function to print the text buffer in reverse order
    void printReverse() {
        if (!tail) {
            cout << "The text buffer is empty!" << endl;
            return;
        }

        TextNode* temp = tail;
        cout << "Text buffer (reverse): ";
        while (temp) {
            cout << temp->character;
            temp = temp->prev;
        }
        cout << endl;
    }
};

int main() {
    TextEditor editor;

    // Insert text
    editor.insertText("Hello, world!");

    // Display text
    editor.displayText();

    // Search for text
    editor.searchText("world");

    // Delete last 6 characters
    editor.deleteText(6);

    // Display updated text
    editor.displayText();

    // Print text in reverse
    editor.printReverse();

    return 0;
}
```

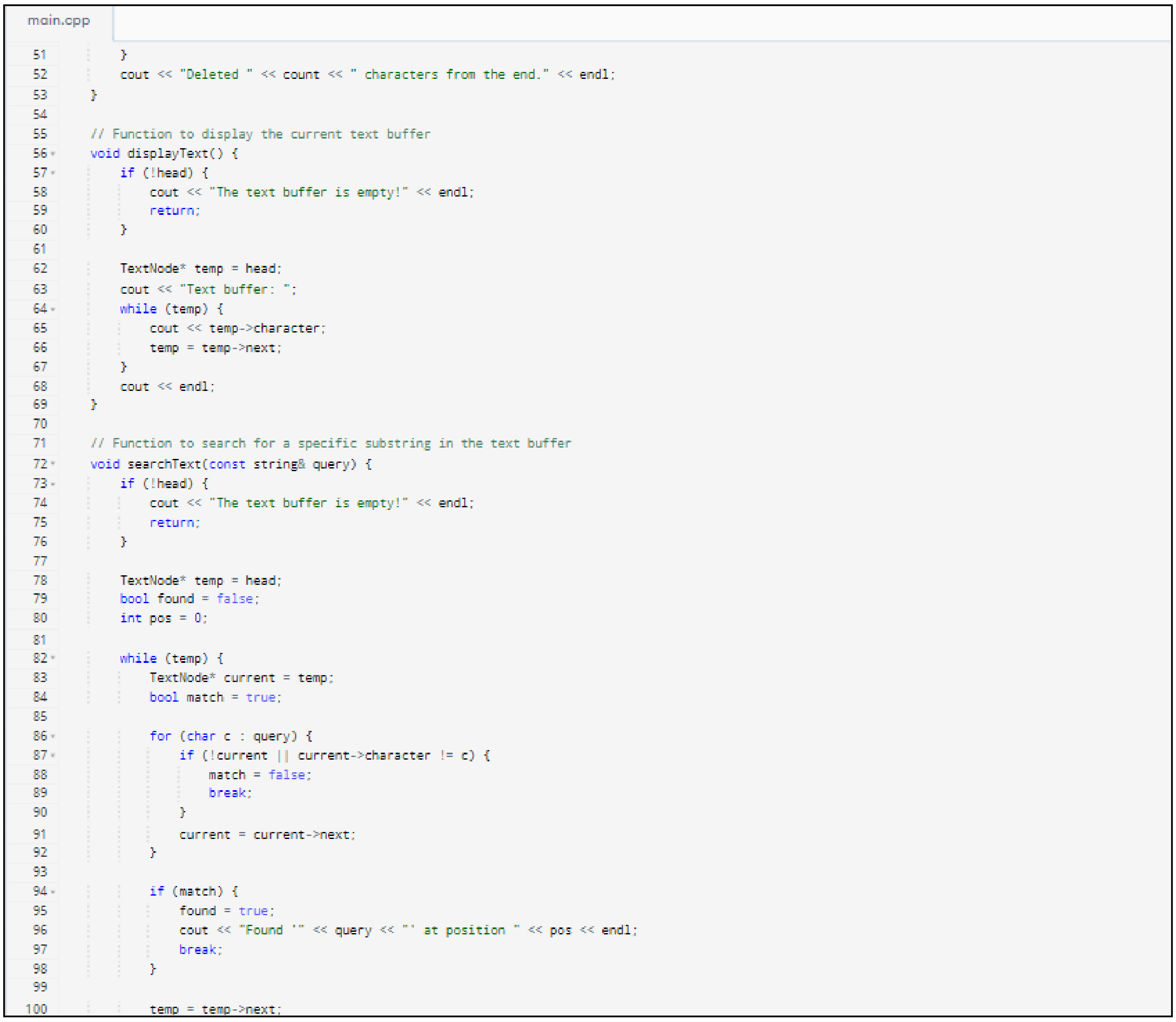
Output :

```
Text inserted: Hello, world!
Text buffer: Hello, world!
Found 'world' at position 7
Deleted 0 characters from the end.
Text buffer: Hello,
Text buffer (reverse):  ,olleH
```

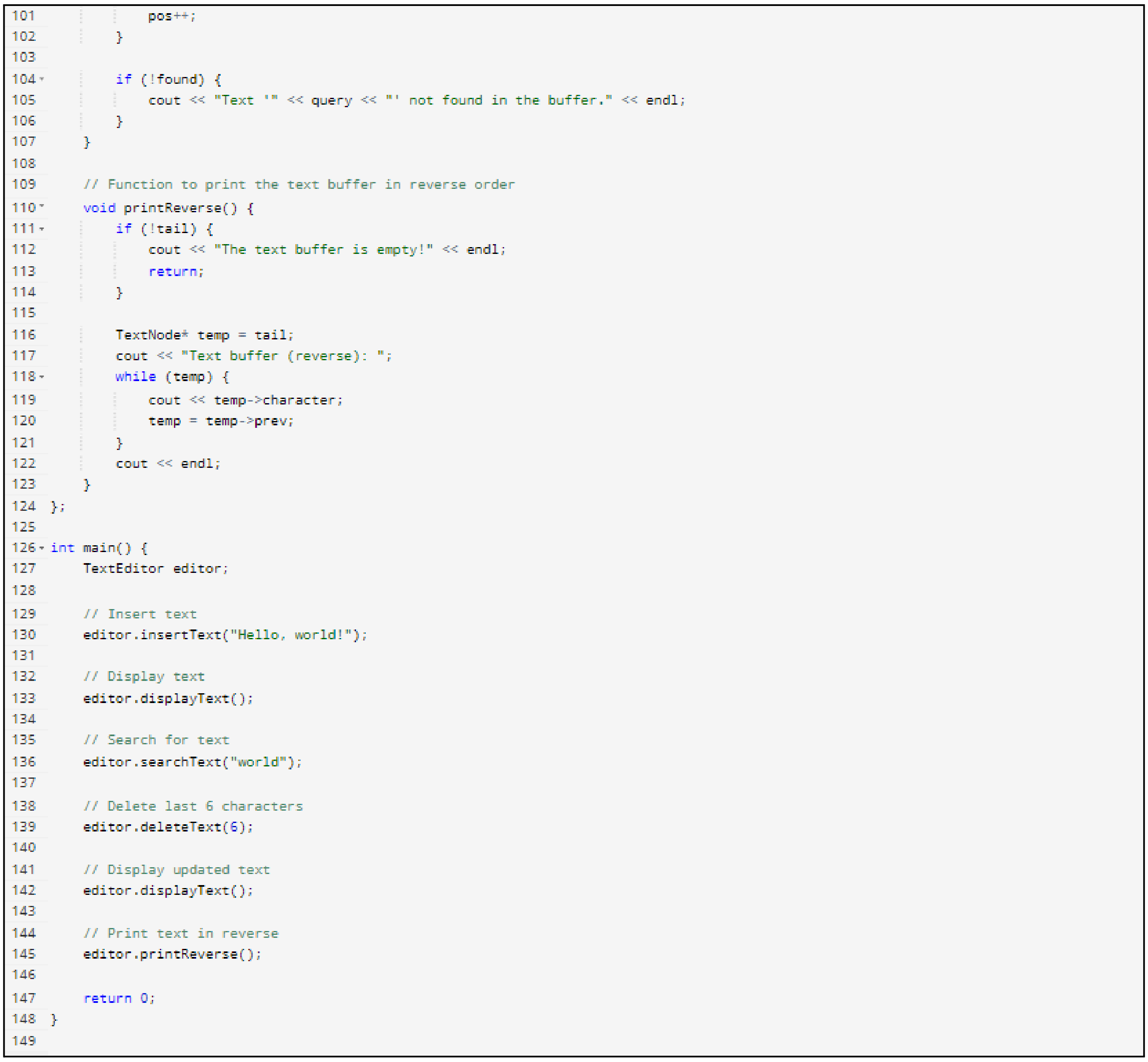


main.cpp

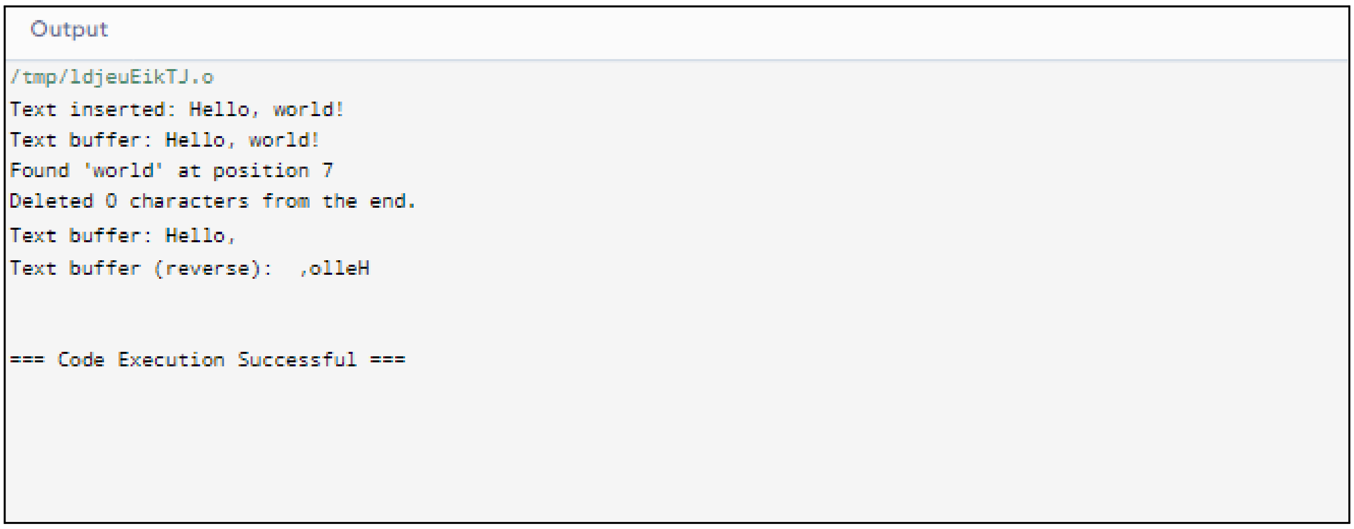
```cpp
#include <iostream>
#include <string>
using namespace std;

// Node structure for each character in the text buffer
class TextNode {
public:
    char character;
    TextNode* prev;
    TextNode* next;

    TextNode(char c) : character(c), prev(nullptr), next(nullptr) {}
};

// TextEditor class using a doubly linked list
class TextEditor {
private:
    TextNode* head;
    TextNode* tail;

public:
    TextEditor() : head(nullptr), tail(nullptr) {}

    // Function to insert text at the end of the buffer
    void insertText(const string& text) {
        for (char c : text) {
            TextNode* newNode = new TextNode(c);
            if (!head) {
                head = tail = newNode;
            } else {
                tail->next = newNode;
                newNode->prev = tail;
                tail = newNode;
            }
        }
        cout << "Text inserted: " << text << endl;
    }

    // Function to delete text from the end of the buffer
    void deleteText(int count) {
        while (count > 0 && tail) {
            TextNode* temp = tail;
            tail = tail->prev;
            if (tail) {
                tail->next = nullptr;
            } else {
                head = nullptr;  // If the list is empty after deletion
            }
            delete temp;
            count--;
```



```cpp
    }
    cout << "Deleted " << count << " characters from the end." << endl;
}

// Function to display the current text buffer
void displayText() {
    if (!head) {
        cout << "The text buffer is empty!" << endl;
        return;
    }

    TextNode* temp = head;
    cout << "Text buffer: ";
    while (temp) {
        cout << temp->character;
        temp = temp->next;
    }
    cout << endl;
}

// Function to search for a specific substring in the text buffer
void searchText(const string& query) {
    if (!head) {
        cout << "The text buffer is empty!" << endl;
        return;
    }

    TextNode* temp = head;
    bool found = false;
    int pos = 0;

    while (temp) {
        TextNode* current = temp;
        bool match = true;

        for (char c : query) {
            if (!current || current->character != c) {
                match = false;
                break;
            }
            current = current->next;
        }

        if (match) {
            found = true;
            cout << "Found '" << query << "' at position " << pos << endl;
            break;
        }

        temp = temp->next;
```



```cpp
            pos++;
        }

        if (!found) {
            cout << "Text '" << query << "' not found in the buffer." << endl;
        }
    }

    // Function to print the text buffer in reverse order
    void printReverse() {
        if (!tail) {
            cout << "The text buffer is empty!" << endl;
            return;
        }

        TextNode* temp = tail;
        cout << "Text buffer (reverse): ";
        while (temp) {
            cout << temp->character;
            temp = temp->prev;
        }
        cout << endl;
    }
};

int main() {
    TextEditor editor;

    // Insert text
    editor.insertText("Hello, world!");

    // Display text
    editor.displayText();

    // Search for text
    editor.searchText("world");

    // Delete last 6 characters
    editor.deleteText(6);

    // Display updated text
    editor.displayText();

    // Print text in reverse
    editor.printReverse();

    return 0;
}
```



Output

```
/tmp/ldjeuEikTJ.o
Text inserted: Hello, world!
Text buffer: Hello, world!
Found 'world' at position 7
Deleted 0 characters from the end.
Text buffer: Hello,
Text buffer (reverse):  ,olleH


=== Code Execution Successful ===
```