

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;
            }
        }
    }
};
```

```

        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

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Node structure for each character in the text buffer
6 class TextNode {
7 public:
8     char character;
9     TextNode* prev;
10    TextNode* next;
11
12    TextNode(char c) : character(c), prev(nullptr), next(nullptr) {}
13 };
14
15 // TextEditor class using a doubly linked list
16 class TextEditor {
17 private:
18     TextNode* head;
19     TextNode* tail;
20
21 public:
22     TextEditor() : head(nullptr), tail(nullptr) {}
23
24     // Function to insert text at the end of the buffer
25     void insertText(const string& text) {
26         for (char c : text) {
27             TextNode* newNode = new TextNode(c);
28             if (!head) {
29                 head = tail = newNode;
30             } else {
31                 tail->next = newNode;
32                 newNode->prev = tail;
33                 tail = newNode;
34             }
35         }
36         cout << "Text inserted: " << text << endl;
37     }
38
39     // Function to delete text from the end of the buffer
40     void deleteText(int count) {
41         while (count > 0 && tail) {
42             TextNode* temp = tail;
43             tail = tail->prev;
44             if (tail) {
45                 tail->next = nullptr;
46             } else {
47                 head = nullptr; // If the list is empty after deletion
48             }
49             delete temp;
50             count--;
```

main.cpp

```
51     }
52     cout << "Deleted " << count << " characters from the end." << endl;
53 }
54
55 // Function to display the current text buffer
56 void displayText() {
57     if (!head) {
58         cout << "The text buffer is empty!" << endl;
59         return;
60     }
61
62     TextNode* temp = head;
63     cout << "Text buffer: ";
64     while (temp) {
65         cout << temp->character;
66         temp = temp->next;
67     }
68     cout << endl;
69 }
70
71 // Function to search for a specific substring in the text buffer
72 void searchText(const string& query) {
73     if (!head) {
74         cout << "The text buffer is empty!" << endl;
75         return;
76     }
77
78     TextNode* temp = head;
79     bool found = false;
80     int pos = 0;
81
82     while (temp) {
83         TextNode* current = temp;
84         bool match = true;
85
86         for (char c : query) {
87             if (!current || current->character != c) {
88                 match = false;
89                 break;
90             }
91             current = current->next;
92         }
93
94         if (match) {
95             found = true;
96             cout << "Found '" << query << "' at position " << pos << endl;
97             break;
98         }
99
100         temp = temp->next;
```

```

101         pos++;
102     }
103
104     if (!found) {
105         cout << "Text '" << query << "' not found in the buffer." << endl;
106     }
107 }
108
109 // Function to print the text buffer in reverse order
110 void printReverse() {
111     if (!tail) {
112         cout << "The text buffer is empty!" << endl;
113         return;
114     }
115
116     TextNode* temp = tail;
117     cout << "Text buffer (reverse): ";
118     while (temp) {
119         cout << temp->character;
120         temp = temp->prev;
121     }
122     cout << endl;
123 }
124 };
125
126 int main() {
127     TextEditor editor;
128
129     // Insert text
130     editor.insertText("Hello, world!");
131
132     // Display text
133     editor.displayText();
134
135     // Search for text
136     editor.searchText("world");
137
138     // Delete last 6 characters
139     editor.deleteText(6);
140
141     // Display updated text
142     editor.displayText();
143
144     // Print text in reverse
145     editor.printReverse();
146
147     return 0;
148 }
149

```

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 ===

```