# **README for TextEditor Implementation**

## **Overview**

The provided code implements a basic text editor with features inspired by traditional command-line text editors like vi or vim. It supports modes (insert and normal), cursor movement, line and character manipulations, and command-based operations. The TextEditor class is the core of the implementation, while the CommandMode class manages command history.

## **Key Features**

### **1. Insert Character**

The insert(char ch) method allows inserting a character at the cursor's position in the current line. If the character count exceeds 30 in a line, the method automatically splits the line into a new one.

#### **Key Functionalities:**

* Inserts the character at the cursor.
* Adjusts links in the doubly linked list of nodes.
* Automatically creates a new line if a line exceeds 30 characters.
* Updates the editor status after every insertion.

### **2. Cursor Movement**

The cursor can be moved around the text using the following methods:

* moveUp() and moveDown() to navigate between lines.
* moveRight() and moveLeft() to navigate within a line.
* moveToStartOfLine() and moveToEndOfLine() to jump to the start or end of the line.
* moveToNextWord() and moveToPreviousWord() for word-based navigation.

### **3. Line Operations**

#### **New Line**

* newLine(): Inserts a new empty line below the current line and moves the cursor to the new line.

#### **Delete Line**

* deleteToEndOfLine(): Deletes all characters from the cursor position to the end of the current line.
* deleteCharacterAtCursor(): Deletes the character at the cursor position and updates the linked list.
* backspace(): Deletes the character before the cursor.

### **4. Clipboard Operations**

The text editor supports yanking (copying) and pasting lines:

* yankLine(): Copies the current line into a buffer.
* pasteAfter() and pasteBefore(): Paste the copied line after or before the current line.

### **5. Mode Handling**

The editor has two modes:

* **Insert Mode:** Allows editing and inserting text.
* **Normal Mode:** Allows command execution and navigation.

Methods for mode management include:

* enterInsertMode() and exitInsertMode() to toggle between modes.
* isInsertMode() to check the current mode.

### **6. Search and Replace**

Commands can be executed in normal mode to search or replace text patterns:

* /pattern: Searches for a specific text pattern.
* n and N: Navigate to the next or previous search result.
* :s/old/new/g: Replaces all occurrences of "old" with "new" in the current line.

### **7. Command Mode**

The CommandMode class manages the history of commands entered by the user. It supports:

* Storing commands in a history list.
* Navigating through the history using getPreviousCommand() and getNextCommand().

### **8. File Operations**

The editor allows saving and loading files:

* :w: Saves the current content to a file.
* :wq: Saves the file and quits the editor.
* :q or :q!: Quits the editor, with or without saving.

### **9. Display and Status Updates**

* The display() method renders the current text in the console, highlighting the cursor.
* The updateStatus() method updates the editor’s status bar to show:
  + Current mode.
  + Cursor position (line and column).
  + Last executed command.

## **Usage Instructions**

1. **Compilation**: Compile the code using a C++ compiler (e.g., g++ -o editor editor.cpp).
2. **Execution**: Run the compiled program (./editor).
3. **Modes**:
   * Use i to enter insert mode for typing text.
   * Use Esc to exit insert mode and return to normal mode.
4. **Commands**:
   * :w to save the file.
   * :q to quit the editor.
   * /pattern to search for a pattern.
5. **Navigation**: Use arrow keys for cursor movement or h, j, k, l for left, down, up, and right.

## **Dependencies**

* The code uses platform-specific functionality for getChar() (e.g., getch() on Windows).
* Ensure platform compatibility by modifying the getChar() implementation as necessary.

## **Future Improvements**

* Implement undo/redo functionality.
* Enhance search and replace to span across multiple lines.
* Add support for mouse-based navigation.
* Introduce syntax highlighting for different file types.

This README serves as a guide to understanding and using the text editor code effectively