LineEd – Your own line editor

1. **Problem Statement**

Develope your own line editor “LineEd” to create text file line by line. When LineEd starts it should display message ‘Enter a command’ and wait for user to LineEd command. After user enters a command LineEd would perform editing on current text buffer according to command and display appropriate output message in case of successful completion of command or error message in case command fails.

LineEd maintain a text buffer of characters entered in form of array of lines. Each line ends with “\0”. It also maintain current line number.

This text buffer gets filled with content when file is read. User can save text buffer into a text file. The default extension of files is .txt so in case file extension is not specified by user it is taken as .txt.

1. **Functional Specification**
2. This editor must give output for every command as it works in actual editor.
3. Accepting input should take care for case sensitivity of characters
4. Make sure your application uses File handling
5. Use Data structures like linked-list, sorting, searching wherever required

**3. Technical Specification**

Following is a list of functionalities of the system. Wherever, the description of functionality is not adequate; you can make appropriate assumptions and proceed.

* + 1. Following commands should be supported by LineEd -
       1. Enter a command - read <filename> Arguments
          1. filename data type is string denotes name of file which is to be read Function -

This command reads file with given filename and loads its content in text buffer. If there is any text in text buffer it is overwritten. Current line number = 1.

Example 1-

Enter a command - read first1

File first1.txt read successfully into buffer.

{ This will read content of file “first1.txt” into text buffer}

Example 2 -

Enter a command - read d:/mydir/message.out

File d:/mydir/message.out read successfully into buffer.

{ This will read content of file “message.out” from d:/mydir into text buffer}

Example 3 -

Enter a command - read noname Error: File noname.txt does not exist

{Assumption: The file noname.txt is not present in the current directory}

* + - 1. Enter a command - save <filename> Arguments
         1. filename data type is string denotes name of file where text buffer is stored. Function -

This command saves text buffer as file. The command should display error if file already exist and should not overwrite file. Current line number does not change.

Example 1-

Enter a command - save myfile

{ This will save content of text buffer as myfile.txt} Example 2 -

Enter a command - save d:/mydir/message.out

{This will save content of text buffer “message.out” from d:/mydir} Example 3 -

Enter a command - save existingFile.inp Error: Unable to write existingFile.inp

{Assumption: The file existingFile.inp is present in the current directory}

* + - 1. Enter a command - append <filename> Arguments
         1. filename data type is string denotes name of file which is to be appended Function -

This command reads file with given filename and appends its content to existing text buffer at the end. If there is no text in text buffer then this command is same as read command. Current line number = 1.

Example 1-

Enter a command - append first1

File first1.txt appended successfully into buffer.

{ This will append content of file “first1.txt” into text buffer}

Example 2 -

Enter a command - append d:/mydir/message.out

File d:/mydir/message.out appended successfully into buffer.

{ This will append content of file “message.out” from d:/mydir into text buffer}

Example 3 -

Enter a command - append noname Error: File noname.txt does not exist

{Assumption: The file noname.txt is not present in the current directory}

* + - 1. Enter a command - display

Display command has various formats

* + - * 1. display with no argument

Displays entire content of text buffer line by line with line numbers. If text buffer is large it may scroll on the screen.

* + - * 1. display noscroll

Displays entire content of text buffer line by line with line numbers. If text buffer is large, it displays 12 lines and wait for user to enter any key. After user enters a key it displays next 12 lines and so on

* + - * 1. display 10-20

Displays entire content of text buffer line by line with line numbers from line 10 to

1. If text buffer contains less than 10 lines then display nothing. If text buffer contains less than 20 lines then display from line 10 till end of text buffer.

Current line number does not change.

Example -

Enter a command - read first1

File first1.txt read successfully into buffer. Enter a command - display

* 1. apple
  2. banana
  3. guava
  4. grape

Enter a command – display 2-3

1. banana
2. guava

Enter a command –

1. replace <oldword> <newword> Arguments -
   1. oldword data type is number denotes word to replace.
   2. newword data type is number denotes word to replace with. Function

This command replace <oldword> with <newword> in the current line. Then displays changed line. Current line number does not change.

Example 1 -

Enter a command - read first1

File first1.txt read successfully into buffer. Enter a command - display

1. apple
2. banana
3. guava
4. grape

Enter a command – goto 4

4. grape

Enter a command – replace grape pear

4. pear

Enter a command – replace abc def

Error: word “abc” not found on current line.

1. add <text> Arguments -
   1. text denotes string of characters. Function -

This command will insert <text> as new line after the current line. Current line number = line number of newly inserted line.

Example -

Enter a command - read first1

File first1.txt read successfully into buffer. Enter a command - display

1. apple
2. banana
3. guava
4. grape

Enter a command – goto 4

4. grape

Enter a command – replace grape pear

1. pear

Enter a command – add bunch of grapes

1. bunch of grapes Enter a command – display
2. apple
3. banana
4. guava
5. pear
6. bunch of grapes
7. find <word> Arguments -
   1. <word> denotes word to find. Function -

This command will find lines containing <word> in text buffer. Current line number does not change.

Example -

Enter a command - read first1

File first1.txt read successfully into buffer. Enter a command - display

1. apple
2. banana
3. guava
4. grape

Enter a command – find banana

2. banana

Enter a command – find pear pear not found

1. delete <FromLineNo>-<toLineNo> Arguments -
   1. <FromLineno> denotes starting line number from where to delete
   2. <ToLineNo> denoted ending line number to delete Function -

This command will delete lines whose line numbers are >= <FromLineNo> and >=

<ToLineNo>. If only <FromLineNo> is given it will delete single line. Current line number

= 1.

Example -

Enter a command - read first1

File first1.txt read successfully into buffer. Enter a command - display

1. apple
2. banana
3. guava
4. grape

Enter a command – delete 2-3 2 lines deleted

Enter a command – display

1. apple
2. grape

Enter a command – delete 1

1 line deleted

Enter a command – display

1. grape

1. quit

No arguments. Function –

This command will quit the LineEd. But before quitting it should check if current contents of text buffer is saved in file. If not, it should ask user if user wants to save the content of text buffer (yes/no), if users chooses ‘yes’ then it should ask filename to save to and save the content of text buffer to that file.