COSC2320: Data Structures

hw2: Recursion with Linked Lists: a Simple Spreadsheet

Updates

1. First update:
   - It is OK to assume that all input numbers will be non-negative.
   - Clarification: when the input text file is empty, you should make an empty linked list and then process operations inside the operation text file (such as possibly add lines, remove lines, and math operations).
   - It is OK to assume that the operation text file will never be empty.
   - Extra credit: operation 'normalize' calculates the average on the indicated line and utilizes it to normalize the whole file (10 points). Example, suppose this is the input file:

```
F       141     151     63      158     Yun     23
F       192     106     92      168     Yun     30
F       107     64      94      224     Yun     54
F       111     72      105     161     Julie   50
F       104     137     118     179     Julie   36
F       119     104     110     137     Julie   103
F       123     113     130     101     Yun     86
T       145     109     89      142     Yun     97
F       194     86      59      92      Yun     84
F       136     106     105     132     Yun     88
F       154     93      87      112     Yun     50
F       132     77      97      92      Yun     96
T       144     103     94      165     Yun     47
```

     And, suppose that the operation file contains this line: `mathline:2:normalize`
     Then, the average of line 2 is: 117.6; now, every numerical value in the input file (actualy the list of lines) needs to be divided by such average. In the example listed, the result would be:

```
F       1       1       0       1       Yun     0
F       1       0       0       1       Yun     0
F       0       0       0       1       Yun     0
F       0       0       0       1       Julie   0
F       0       1       1       1       Julie   0
F       1       0       0       1       Julie   0
F       1       0       1       0       Yun     0
T       1       0       0       1       Yun     0
F       1       0       0       0       Yun     0
F       1       0       0       1       Yun     0
F       1       0       0       0       Yun     0
F       1       0       0       0       Yun     0
T       1       0       0       1       Yun     0
```

   - Expected test-cases point allocation:
     If operation delete does not work: -10
     If operation insert does not work: -10
     If operation count does not work: -20
     If operation max does not work: -20
     If operation min does not work: -20
     If operation mean does not work: -20

1 Introduction

You will create a C++ program that can edit a text file with line and word insertions and deletions, and do some math operations on it. The file consists of a sequence of lines. Each line has words or numbers separated by the TAB

character (that is, \t). Notice that these operations are sufficient to perform basic spreadsheet operations, mimicking how a user manipulates a spreadsheet. The goal is to manipulate the file with linked lists. In particular, the file will be a list of lines, and each line will be a list of words or numbers separated by tabs. Editing the file will be done with a sequence of line insertions, deletions, or math operations.

2 Program input and output specification.

The main program should be called "numbers". The output should be sent to the standard output (via cout or printf, so that it can visualized or redirected to a file). The input file is a regular plain-text file, where each line is terminated by the end-of-line character. There is a second file containing editing operations. Call syntax at the OS prompt:

```
numbers "input=FILENAME1;operations=FILENAME2"
```

Place your codes in a folder named

```
hw2
```

on your home folder. Failure to do so will cause your program to have a zero grade due to inability for automated grading.

Technical details

- The input and operations files are plain-text files, where each line ends with end-of-line character.
- Separator: tab (exactly 1 tab). Separators do not count as words or numbers. Separators are ignored to determine positions.
- Operation 'insertline' inserts a line at the indicated position. The line number must be 1 ... N, where N is the total number of lines. The inserted line must be a blank line that does not have a separator (such as TAB). If an inserted line has a TAB or space, then it may cause test cases to fail.
- Operation 'deleteline' deletes the line of the indicated position.
- Operation 'mathline' performs an operation on the specified line only. The result of the operation is a number that is added to the end of the line. Such result must be preceeded by a TAB, except in the case that the operation was done in an empty line.
-- 'mathline' operation 'count' counts how many numbers there are in a line (it does not count words).
-- 'mathline' operation 'max' finds the maximum value of all numbers in a line.
-- 'mathline' operation 'min' finds the minimum value of all numbers in a line.
-- 'mathline' operation 'mean' finds the mean value of all numbers in a line.
-- The operations max, min, mean will have the value 'undefined' when there are no numbers for them to work on.

- You should verify the correctness of your output with any text editor (such as vi or nano in Unix, notepad in Windows). Note that the line number is not to be shown in the output file (some editors can display line numbers). To show line numbers in Unix use: cat -n filename
- You can assume that all operations will be given in lowercase.
- You can not assume that the operations will come "sorted" by line numbers.
- The use of linked lists and recursive functions are a major requirement. Each line is maintained as a list, where each node will store either a word or a number.
- The file (sequence of lines) can be manipulated as a singly or doubly linked list. A program that uses arrays for either the lines or the content of each line will get 1/2 points (that is, a maximum of 50).
- Memory allocation and deallocation: It is not acceptable to use static memory allocation (such as arrays) or to "forget" to clean memory (via delete). Your program must use 'new' to allocate each node and it must free up the memory before the main() program ends (such as in the last destructor). A program with poor memory management will likely crash, which will be reflected in getting zero points for each test case where this happens.
- You can assume that input text lines can have upto 256 characters, but that should not be a limit in the list.
- The minimum number of lines in an input file is zero (empty). You cannot assume a maximum number of lines in the input file.

Example

Example of program call at the OS prompt: `numbers "input=experiments.txt;operations=work1.txt"`

Input file example:

```
worm    length  px      py
aa      200.001 100     110
ab      220.042 150     100
ac      218.024 250     90
ad      198.035 210     140
```

Example of operations file (note that the operations are not sorted by line number) :

```
deleteline:4
insertline:2
mathline:2:count
mathline:1:count
mathline:2:count
mathline:3:count
mathline:3:count
mathline:4:max
mathline:1:min
mathline:2:min
mathline:5:min
insertline:2
mathline:2:max
mathline:1:mean
mathline:3:mean
mathline:2:count
mathline:2:count
mathline:2:count
mathline:2:mean
```

Output Example:

```
worm    length  px      py      0       0       0
undefined       0       1       2       1
0       1       0       0
aa      200.001 100     110     3       4
ab      220.042 150     100     220
ad      198.035 210     140     140
```

Requirements

- Correctness is the most important requirement. Your program should not crash or produce exceptions. Any unexpected output will affect automated grading and likely this will mean getting zero points on failed test cases.
- All operations should be recursive. Inserting or deleting nodes do not have to be recursive.
-- Using loops (such as 'while', 'for') to manipulate a list are not allowed.
-- Recursive functions are a requirement only for the linked lists, not for the other parts of the program (such as reading the file, and reading parameters).
- Your program should not display the line numbers when producing the final output (if it does then test cases will fail).
- Your program should not produce any output to the console when it is doing intermediate calculations because it will interfere with automated grading and test cases will fail.
- The output of any calculated operation should not have any decimal digits, that is, it needs to be truncated (no rounding). The decimal point should never be in the output of calculated operations. For example, if the values to calculate mean are 1 2 2 then the result of the mean is 5/3, which must be output without rounding and truncating decimals, resulting in: 1