Homework Project 1

Given 2/22/2016, Due 3/08/2016

For a text editor, you need a representation of the text in which you can insert or delete something without copying all characters beyond the insertion. The classical technique for this is to represent the text as a sequence of lines, where the lines are organized in a structure that allows access to the i-th line, or to insert a new line immediately before the i-th line.

So we need a structure with a number, the index of the line, as key, and it returns a line of text, represented by a `char *` object. We can then change this line, and save it in the structure again, and possibly we want to insert a new line immediately before the i-th line, by which the new line becomes the i-th line, and every following line is renumbered. The lines are numbered from 1 to n with an artificial empty line $n + 1$ as end marker.

So our structure should support the following operations

- `text_t * create_text()` creates an empty text, whose length is 0.
- `int length_text( text_t *txt)` returns the number of lines of the current text.
- `char * get_line( text_t *txt, int index)` gets the line of number `index`, if such a line exists, and returns `NULL` else.
- `void append_line( text_t *txt, char * new_line)` appends `new_line` as new last line.
- `char * set_line( text_t *txt, int index, char * new_line)` sets the line of number `index`, if such a line exists, to `new_line`, and returns a pointer to the previous line of that number. If no line of that number exists, it does not change the structure and returns `NULL`.
- `void insert_line( text_t *txt, int index, char * new_line)` inserts the line before the line of number `index`, if such a line exists, to `new_line`, renumbering all lines after that line. If no such line exists, it appends `new_line` as new last line.
- `char * delete_line( text_t *txt, int index)` deletes the line of number `index`, renumbering all lines after that line, and returns a pointer to the deleted line.

You should base your implementation on a balanced search tree, but you must change the key mechanism in such a way that we can easily increase the keys of all leaves above a certain key, without visiting more than $O(\log n)$ nodes. You can use my search tree code.

An implementation as linked list of lines, or any implementation that explicitely renumbers all following lines, is too slow, and will not be accepted.

Submit your code by e-mail to `phjmbrass@gmail.com`. Include the course number I96 and the homework number hw1 in the subject line. Test your code before submission. Do not share code with other students, or use code from the web.