Course CSC 541 Advanced Data Structure Fall Semester 2012
Homework Project 1

Given Aug. 30, 2012, **Due: Sept. 17 Midnight 2012**

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 may use the search tree code (basicsearchtree.c) as a starting point.
<u>An implementation as linked list of lines, or any implementation that explicitly renumbers all following lines, is too slow, and will not be accepted.</u>

Submission instructions:
You need to submit a single .c file through Moodle and rename it as Your_Unity_ID.c.
“test.c” contains all test cases and is available for download through Moodle. Grading will be done using the code in test.c, so you should integrate that into your main function to ensure you have tested your code thoroughly.

Note: **Sharing your code with others will be treated as academic dishonesty and be dealt with very severely.**