Programming Assignment 3: Augmenting Binary Search Trees

Due Monday, March 17 at 10:59AM

In this assignment you will modify the binary search tree code used in lab to support several new features (some with runtime requirements).

These features will require *augmentation* of the existing data structures with additional book-keeping information. This bookkeeping info must be dept up to date incrementally; as a result you will have to modify some existing functions (insert, delete, build-from-array).

Now to the new functions/features:

```
/* allocates an integer array, populates it with the
   elements of t (in-order) and returns the array as an
   int pointer */
extern int * bst_to_array(BST_PTR t);

/* returns the ith smallest element in t.  i ranges
   from 1..n where n is the number of elements in
   the tree.

   If i is outside this range, an error message is printed
   to stderr and the return value is arbitrary (you may return
   whatever you like, but it has no meaning.

   Runtime:  O(h) where h is the tree height
*/
extern int bst_get_ith(BST_PTR t, int i);

/* returns the value in the tree closest to x -- in other
   words, some y in the three where |x-y| is minimum.

   If the tree is empty, a message is sent to stderr and
   the return value is of your choosing.

   Runtime:  O(h) where h is the tree height.
*/
extern int bst_get_nearest(BST_PTR t, int x);
```

```
/* returns the number of elements in t which are greater
   than or equal to x.

   Runtime:  O(h) where h is the tree height
*/
extern int bst_num_geq(BST_PTR *t, int x);

/* returns the number of elements in t which are less
   than or equal to x.

   Runtime:  O(h) where h is the tree height
*/
extern int bst_num_leq(BST_PTR *t, int x);
```

Additional runtime requirement:

Modify the size function to have O(1) runtime.

Functions needing modification:

The insert and remove functions modify the tree. You will need to change them so that they also make sure that the bookkeeping information is correct.

The runtime of these functions must still be O(h)

Submission: you will just submit bst.c

Comments/Suggestions:

You will need to augment the NODE struct. This is perfectly fine since it is in the .c file and not the .h file.

I recommend you write a sanity-checker function which, by brute force, tests whether the bookkeeping information you've maintained is indeed correct.

Of course, you should write extensive test cases. You are free to share test cases with classmates. You might try valgrind to also look for memory errors.