CSE 3100 Systems Programming — Due: 11/22/20 @ 11:55PM

PS # 8

1 Inserting into Binary Search Tree (100 points)

In a past lab, you implemented a binary search tree. We're going to make a multithreaded version of this. The tree, and each node in it, has a mutex associated with it:

```
typedef struct TNode{
  int val;
  struct TNode* left;
  struct TNode* right;
  pthread_mutex_t lock;
} TNode;

typedef struct Tree{
  TNode* root;
  pthread_mutex_t lock;
} Tree;
```

Open `treeMT.c` with a text editor, and implement the `makeEmptyTree` and `insertIntoTree` functions. Use the mutexes to coordinate threads so that they don't interfere with one another.

1.1 `makeEmptyTree`

For this function, allocate memory for a `Tree`, and set the `root` field to `NULL`. Initialize the `lock` field.

1.2 `insertIntoTree`

Traverse the tree until you've found the correct insertion point, and insert the node. You need to use the mutexes to prevent the threads from interfering with one another. Use a fine-grained approach here: as you traverse the tree, hopping from parent node to child node, unlock the parent's mutex when you no longer need it locked (hint: be sure that the child's mutex is locked before unlocking the parent's mutex). This function can either be iterative, or recursive (you chose!). However, if you use recursion, you may find it helpful to create an auxillary function.
Feel free to add anything to the `treeMT.h` file you want.

1.3 Testing

The `treeMTTest.c` file provides the main function. It starts a bunch of threads and which, collectively, insert the numbers 0 through $n-1$ into the tree. Then, it prints out the inorder traversal of the tree, which should be the numbers 0 through $n-1$ in ascending order. You can compile and run this:

```
$ make
gcc -g -Wall treeMT.c -c -o treeMT.o
gcc -g -Wall treeMTTest.c -o treeMTTest treeMT.o -pthread
$ ./treeMTTest 20 3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

The first argument to `treeMTTest` is n. The second argument is the number of threads.
Enjoy!