Threaded Binary Search Trees

CSCI 112, Lab 8

File names: Names of files, functions, and variables, when specified, must be EXACTLY as specified. This includes simple mistakes such as capitalization.

Individual work: All work must be your own. Do not share code with anyone other than the instructor and teaching assistants. This includes looking over shoulders at screens with the code open. You may discuss ideas, algorithms, approaches, *etc.* with other students but NEVER actual code. Do not use code written by anyone else, in the class or from the internet.

Documentation: Each file should begin with a docstring that includes your name, the class number and name, the lab number, and a short description of the lab, as well as documentation pertinent to that particular file.

The project: Add two modifications to the binary search tree of the textbook. Make your modifications as small as possible.

1. Modify the code for a binary search tree to make it threaded. Write a non-recursive inorder traversal method for the threaded binary search tree. A threaded binary tree maintains a reference from each node to its successor.
2. Modify our implementation of the binary search tree so that it handles duplicate keys properly. That is, if a key is already in the tree then the new payload should replace the old rather than add another node with the same key.

Unit tests: The unmodified version of a binary search tree and your modified version should behave the same for any sequence of operations. This should suggest an easy way to create unit tests for every operation.

File names: Call your module `tbst.py` and your unit test module `tbst_test.py`, place in a folder `csci112lab08yourname` zip and turn in to canvas.