7.22. Programming Exercises

1. Extend the `buildParseTree` function to handle mathematical expressions that do not have spaces between every character.

2. Modify the `buildParseTree` and `evaluate` functions to handle boolean statements (and, or, and not). Remember that "not" is a unary operator, so this will complicate your code somewhat.

3. Using the `findSuccessor` method, write a non-recursive inorder traversal for a binary search tree.

4. 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.

5. 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.

6. Create a binary heap with a limited heap size. In other words, the heap only keeps track of the `n` most important items. If the heap grows in size to more than `n` items the least important item is dropped.

7. Clean up the `printexp` function so that it does not include an 'extra' set of parentheses around each number.

8. Using the `buildHeap` method, write a sorting function that can sort a list in $O(n \log n)$ time.

9. Write a function that takes a parse tree for a mathematical expression and calculates the derivative of the expression with respect to some variable.

10. Implement a binary heap as a max heap.

11. Using the `BinaryHeap` class, implement a new class called `PriorityQueue`. Your `PriorityQueue` class should implement the constructor, plus the `enqueue` and `dequeue` methods.

(DiscussionQuestions.html) (../Graphs/toctree.

user not logged in

(DiscussionQuestions.html) (../Graphs/toctree.