Blog Post Sample 1

- `binary_search_tree` is used as an English phrase rather than a class or function name
  - fix: use "Binary Search Tree"
- "efficient" how?
  - fix: "efficient to look up data"
- "connect to one another" what does "connect" mean?
  - Remove that information, start by defining nodes first?
- "branches out in 2 directions" what is this notion of directionality? The text makes sense after the fact, but not while the reader is reading left-to-right
  - Start by defining nodes first, and what a node contains
- what is the relationship between the "data" and the "index" or "key"?
- how do we compare a "key" with a "node"?
  - fix: whose key is less than that of the root
- is an "item" a piece of data, or a key?
- "figure out where to go" what does this mean? what are the choices in the direction?
  - fix: determine whether to search the`left_branch` or the `right_branch`
- "item" has a key (node should have the key?)
  - define how to obtain the key
- what exactly is called a "balanced tree"?
  - A tree with this property is called a balanced tree.
  - Better yet, "To maximize efficiency, we can build a *balanced tree*: a tree where. . . "
- "middle"/"midpoint"/"begininng"/"end": what ordering?
  - fix: use the words "smallest" and "largest" throughout (or least/greatest)
- I don't understand that last sentence at all. . .

Blog Post Sample 2

This one has a lot more grammatical issues

- "may be asked. . . " passive voice
  - "someone who is trying to . . . may ask"
- unclear **why** someone might want to use a binary search tree rather than a list
- description of the "node" is better here
- misuse of ";" in the last sentence of the second paragraph
- the first "," in the third paragraph should be a ":"
- "new value": is this a value to be searched, or a value to be added?
  - actually, this blog post haven't really talked about the ordering of the nodes!
- last paragraph "binary search tree" should be quoted
- "dividing the problem in half" how?
  - fix: Binary search is an algorithm used to find a value in a sorted list. The algorithm iteratively checks the midpoint of a portion of the list, and. . .
- "look at" imprecise
  - fix: compare the value to the left and right child. . .
- phrases like "to keep it simple" and "to put simply" don't really add much