Intersect Design Document

Program Flow

1. Check for valid amount of arguments
2. Verify files were able to open
3. Print to STDERR if files can't be opened
4. Add everything in the first file to the tree
5. Read word at a time from all other files
6. Mark in the tree if that word has been seen before
7. Remove any nodes in the tree that do not have marks
8. Read in next file
9. Print the tree

Notable Data Structures

- Struct tree
- Struct node

Notable Functions

- Void treeAddWords(tree **t, char *line);
    - Adds words into the tree
- void treeIntersects(tree **t, FILE *fp, size_t index);
    - Reads in words of a file and marks in the tree if it has seen it, then deletes unseen nodes

Anticipated Challenges

1. Rebalancing
2. Marking Repeat Words
3. UTF-8 compares
4. Deleting nodes

Targeted Features

1. UTF-8 Support
2. Sort by Ascii
3. Read from STDIN if entering – as the first argument

Architecture

Using a tree to store the initial file's data. The initial tree creation might take awhile but after it is created, all searches on the tree will be log(n). Also, reading in every file after the first one will be done one word at a time.