Helen See
HyunMo Yang

Memory:

We did not use any arrays to hold information. Malloc was used for two buffers, buffer for user input and buffer for reading the file. All the nodes for our structs had memory allocated with calloc. Every malloc and calloc were freed. Even though the memory was dynamically allocated, buffer still uses 1024 char space. That does mean a lot of memory is being taken by buffer, but all of it is freed.

Structure:

There are two linked lists. One linked list contains the users search word(s). The other linked list contains the file names and occurrences. Occurrences is needed for search AND. Occurrences will be an indicator of which file names to print. If there are five words to search and one file has five occurrences, we know that the file contains all of the user word searches.

Implementation:

In the function, setWordSearch, the user inputs either so, sa, or q and the words. It splits the input with strtok() into one of the structures. As we store, duplicate tokens are turned away. The first token is determined to be sa, so, or q. With command sa, the amount of tokens the user inputted, not including sa, is counted. For both so and sa, we search the file for the words. Once found, the file names are stored into our other linked list structure. Command so prints every file name in the linked list. Command sa prints the file names only if the occurrence is equivalent to the total amount of words.

Indexer Format:

The same specs as pa3. The file is organized with:

```
 <list> token
filenames occurrences filenames occurrences …...
</list>
```