Eric Bronner && Tobias Perelstein
CS214 – Brian Russell
Search – pa4
Readme

For our approach to this project, we begin by reading and parsing the input file. The input file is read line by line using the getline() function and a buffer, and then each line is tokenized using the strtok function. We create a hash table using UTHash, with each key returning a pointer to the head of a SortedList from pa2. If the token is <list>, it is clear that the next token is the key, which is saved in a different string. Until the token is </list>, it is clear that all other tokens are file names, which are added to the SortedList. After, the file is closed and the buffer is freed.

After the tokens are successfully added to the hash table, the user is prompted for a query or to quit the program. The input is scanned in and checked to see if it is 'q' or 'Q', in which case the program gracefully shuts itself down, freeing the query string. Otherwise, the first two characters of the input are checked against “so” and “sa”; if it is one of them, the proper search function is called, otherwise the user is asked to enter a properly formatted query.

The so search tokenizes the user input, ignoring the first term (which is still “so”) and returning an error if there is no other term. If a string cannot be found in the hash table, the function ignores it. If all is good, a new SortedList is created, and every file name found from the hash table searches is added to the list (which does not add duplicates). After, the list is printed, and then destroyed.

The sa search copies the first token's SortedList of file names, and then searches through the result and each next token's SortedList of file names. If a file name from the result list is not found in a later token's list, that node is flagged as removed. After all inputs are checked, the result list is printed, except for flagged nodes. Then, the entire list is freed.