```
/***
 *CS214 Assignment 2: Recursive Indexing
 *due 3/29
 *
 *@author Andrew Pagano
 *@author Thurgood Kilper
 */

Time Complexity:O(m+n)=(find #tokens + #files) +
  O(n^2)=(insert token and filename into two linked lists) +
  O(n)=(print contents two linked lists) +
  O(n)=(frees both linked lists) =
  O(n^2)

Space Complexity: O(n)=(# of tokens + filenames turned into nodes in lists) +
    O(1)=(recursive directory calls) +
    O(n)=(file calls) =
    O(n)

Design: The main function starts by using the stat struct to figure out if the given
input is a file or directory and then calls either openFile() or openDirectory() on
argv[2]. openDirectory is a void function with char* parameter. Given the name of a
directory, it will opendir() if valid, attach the path to the beginning, and either
recursively call openDirectory() or call openFile() until there is nothing left.
openFile is a void function with char* parameter. Given the name of a file, it will
open the file with fopen(), extract a string of the entire contents, find each
token, call tolower() on each char of each token, insert the token into the master
linked list with stringInsert(), and then free each token and the string.
stringInsert() sorts and inserts each unique token into the linkedlist, and creates
a linked list for each token which holds the file names which it is found in by
listInsert(). Once all files and directories are read, the main will call printList()
which prints each token and corresponding file names in xml format. printList() will
create a new file and argv[1], write() the linked lists in the correct format.
The final step is to cleanList() by freeing all of the tokens and file names in the
linked lists.
```