Assignment 7 - The Great Firewall of Santa Cruz

Description: This program will implement a firewall that will filter out certain words that are in text files and replace them with new words using hash tables. The words that will be replaced will be taken in from another text file and stored in a hashtable and then implemented into the Bloom filter that will parse through a given text file.

banhammer.c: This contains main() and may contain the other functions necessary to complete the assignment.
messages.h: Defines the mixspeak, badspeak, and goodspeak messages that are used in banhammer.c.
salts.h: Defines the primary, secondary, and tertiary salts to be used in your Bloom filter implementation. Also defines the salt used by the hash table in your hash table implementation.
speck.h: Defines the interface for the hash function using the SPECK cipher.
speck.c: Contains the implementation of the hash function using the SPECK cipher.
ht.h: Defines the interface for the hash table .
ht.c: Contains the implementation of the hash table .
bst.h: Defines the interface for the binary search tree .
bst.c: Contains the implementation of the binary search tree .
node.h: Defines the interface for the node .
node.c: Contains the implementation of the node .
bf.h: Defines the interface for the Bloom filter .
bf.c: Contains the implementation of the Bloom filter .
bv.h: Defines the interface for the bit vector .
bv.c: Contains the implementation of the bit vector .
parser.h: Defines the interface for the regex parsing module.
parser.c: Contains the implementation of the regex parsing module.
Makefile: This is a file that will allow the grader to type make to compile your program.
• CC = clang must be specified.
• CFLAGS = -Wall -Wextra -Werror -Wpedantic must be included.
• make should build the banhammer executable, as should make all and make banhammer.
• make clean must remove all files that are compiler generated.
• make format should format all your source code, including the header files.
README.md: This must be in Markdown. This must describe how to use your program and Makefile. This includes listing and explaining the command-line options that your program accepts.

DESIGN.pdf: This must be a PDF. The design document should describe your design for your program with enough detail that a sufficiently knowledgeable programmer would be able to replicate your implementation.

WRITEUP.pdf: This document must be a PDF. The writeup must include at least the following:
• Graphs comparing the total number of lookups and average binary search tree branches traversed as you vary the hash table and Bloom filter size.
• Analysis of the graphs you produce

Pseudocode
bv.c
bv_create(length)
Creates a bit vector with a given length and dynamically allocated 8 bit vector.
bv_delete()
Frees vector and bit vector
bv_length returns length
Bv_set_bit()
sets bit of vector to 1 using bitwise equations
bv_clr_bit
Clears bit of vector to 1 using bitwise equations
Bv_get_bit return the bit and specified index of vector
bf.c
BloomFilter *bf_create(uint32_t size)
    Adds the salts to primary,secondary,and tertiary uint32_t[2] arrays through salts.h
    And creates a bit vector as filter
Bv_delete
Frees dynamically allocated arrays,filter, and bf
bf_size(returns bit vector length)
bf_insert(oldspeak)
Hashes 3 salts with oldspeak for their indexes and sets the bits
bf_probe(oldspeak)
    Returns true if oldspeak has been added to bloom filter
bf_count()
Counts the number of initialized indexes

node.c
node_create(old,new)

Creates a node with char pointers old speak and newspeak and empty left and right nodes
node_delete frees node but not children nodes
node_print()
Prints oldspeak and newspeak

bst.c
Bst_create returns NULL
Bst_height returns the height of the bst
Bst_size returns the total number of nodes in the tree
bst_find(returns the node according to the oldspeak)
bst_insert(adds node based on alphabetical ordering)
bst_delete(recursively postorder deletes nodes)
bst_print(prints out bst in order)

ht.c
ht_create(creates the salt for hash and dynamically allocates an array of bst and stores size
ht_delete()deletes bst trees and frees ht
Ht_size returns size
ht_lookup()uses bst_find to determine if oldspeak is in ht
ht_insert()inserts a new node based off oldspeak and newspeak using hashing to find the index
ht_count()counts the number of non-Null bsts
ht_avg_bst_size()returns the average bst size
ht_avg_bst_height()returns the average bst height
banhammer.c
Creates getopt command line options for the bloom filter and hash table sizes, help command, and stats
determines if regex is valid which is supposed to encompass alphabetical letter 0-9 hyphens, apostrophes and underscores.
parses through stdin and checks if the word is part of badspeak and newspeak and prints them out accordingly by storing them into a bst for old message and one for new message
Prints out stats if enabled
Prints out message.h message according to crime and printed out from bad message and new message bst
Free and close all files and functions.
Initializes BloomFilter and HashTable
Opens badspeak and newspeak txt

scans in all the bad speak into the bf and ht
scans in all the newspeak into the bf and ht

Notes:
Used Code provided by Professor Long in assignment doc and lecture 18 for bst.c
Watched Euguene's section for pseudocode