Assignment 7: The Great Firewall of Santa Cruz: Bloom Filters, Linked Lists, Binary Trees and Hash Tables
This program can scan through user inputted text to detect the use of banned words, notifying the user when they use either oldspeak or badspeak.

banhammer.c:
The main file that handles all interactions with the user, mostly makes use of the various functions found within the other files to properly scan text and detect oldspeak/badspeak.
Can be launched with any of 5 different command options.
-t: specifies the size of the hash table
-f: specifies the size of the Bloom filter
-s: prints various program statistics to stdout
-h: displays program usage
Program first scans through a list of badspeak words and another list of oldspeak/newspeak words, filling a hashtable.
Scan through a list of words to see evaluate if it has been added to the bloom filter, if it has been then it is either a badspeak or an oldspeak.
If oldspeak is detected it sends the user a message regarding which oldspeak words can be replaced with newspeak words.
If badspeak is detected a message is sent saying that the user will be sent to joycamp.

bf.c:
File that stores the bloom filter ADT. Largely just makes use of the bitvector ADT and some hash functions that make use of salts.h.
-can create and delete a bloom filter object which consits of three salt arrays and a bitvector
bf_insert: inserts into the bitvector object after hashing the requested oldspeak with all three of the salts
bf_probe: scans through the bitvector at the three bits designated by the hashed salts, if all three scanned bits are 1, then the oldspeak was added to the bloom filter

bst.c:
File that stores the binary search tree ADT. Effectively just used as a way to efficiently store and retrieve nodes. Notably there is no actual struct, instead all functions are simply used for the logic of traversing interconnected node s.
-bst_create: literally does nothing apart from return NULL
-bst_delete: conducts a postorder traversal on the tree, calling node_delete on each node
-bst_height: scans through the tree to calculate the height of the tree
-bst_size: counts the number of nodes located within the tree
-bst_find: attempts to find the requested when given the node's oldspeak by scanning through the tree
-bst_insert: inserts a given node into a tree in the correct position, ignores duplicate nodes

bv.c:
File that stores the bitvector ADT, used largely only by the bloomfilter ADT.
Literally just a copy of the Code ADT in assignment 6.
-has the ability to create and delete a bitvector
-can set and clear a bit at a specific position
-can also return the value of a bit at a specific position

ht.c:
File that stores the hashtable ADT, largely just makes use of the binary search tree ADT.
-can create and delete a hashtable
ht_lookup: uses bst_find to attempt to find a requested node at the binary search tree specified by the hashed value of the oldspeak
ht_insert: uses bst_insert to insert a specified node into the bst specified by the hashed value of the oldspeak
-can return the size of the hashtable, the average bst size and the average bst height

node.c:

File that stores the Node ADT used for the binary search tree ADT. Only has three functions.
Does not do much on its own, only really used by the other helper files.
-node_create: creates a node and fills it with the user inputted oldspeak and newspeak
-node_delete: frees the node's memory, notably does not impact left/right nodes