Assignment 6
The Great Firewall of Santa Cruz:
Bloom Filters, Linked Lists and Hash Tables

Pre-lab Part 1

1. Write down the pseudocode for inserting and deleting elements from a Bloom filter.

function ***bf_insert***(filter, oldspeak):

index1 = hash(filter.primary_hash, oldspeak) % 2 filter.length

index2 = hash(filter.secondary_hash, oldspeak) % 2 filter.length

index3 = hash(filter. tertiary_hash, oldspeak) % 2 filter.length

set bit at index1

set bit at index2

set bit at index3

function ***bf_probe***(filter, oldspeak):

index1 = hash(filter.primary_hash, oldspeak) % filter.length

index2 = hash(filter.secondary_hash, oldspeak) % filter.length

index3 = hash(filter. tertiary_hash, oldspeak) % filter.length

if all three bits at index1, index2 and index3 are set then return TRUE

else return FALSE

Pre-lab Part 2

1. Write down the pseudocode for each of the functions in the interface for the linked list ADT.

function ***ll_create***(mtf):

let ll is pointer to new list

allocate memory for ll

create head and tail nodes

```
        head.prev = tail.next = NULL
        head.next = tail
        tail.prev = head
        set ll length to 0
        set ll.mtf to mtf
        return ll

function ll_delete(ll):
        let node n is ll.head.next
        WHILE n != ll.tail do:
                deleted_node = n
                n = n.next
                node_delete(deleted)

        node_delete(ll.head)
        node_delete(ll.tail)
        free(ll)
        ll = NULL

function ll_length(ll):
        return ll.length

function ll_lookup(ll, oldspeak):
        let node n is ll.head.next
        WHILE n != ll.tail do:
                If n.oldspeak == oldspeak:
                        If ll.mtf == TRUE:
                                Remove node n from the list and re-insert it at the head of the list
                        return n
```

```
            n = n.next
        return NULL

function ll_insert(ll, oldspeak, newspeak):
        if ll_lookup(ll, oldspeak) == TRUE then return
        newnode = node_create(oldspeak, newspeak)
        ll.head.next.prev = newnode
        newnode.next = ll.head.next
        newnode.prev = ll.head
        ll.head.next = newnode
        ll.length += 1

function ll_print(ll):
        let node n is ll.head.next
        WHILE n != ll.tail do:
            node_print(n)
            n = n.next
```

Pre-lab Part 3

1. Write down the regular expression you will use to match words with. It should match hyphenations and concatenations as well.

"[a-zA-Z0-9_\-\']+"

Program Pseudocode

let **h_size** = 10000

let **f_size** = 2 ^ 20

let **mtf** = FALSE

Parse command line.

If the "-h" option is specified then set **h_size** to option argument

If the “-f” option is specified then set **f_size** to option argument

If the “-m” option is specified then set **mtf** to TRUE

Create new HashTable **ht** of **h_size**

Create new BloomFilter **bf** of **f_size**

Read “badspeak.txt”

Open “badspeak.txt” file

For each **word** in the badspeak file:

    Insert **word** into **ht** (with newspeak equals to NULL)

    Insert **word** into **bf**

Read “newspeak.txt”

Open “newspeak.txt” file

For each word pair { **oldspeak** , **newspeak** } in the newspeak file:

    Insert **oldspeak** -> **newspeak** pair into **ht** (with newspeak equals to NULL)

    Insert **oldspeak** into **bf**

Let **bad_speak_list** = ***ll_create***(false)

Let **new_speak_list** = ***ll_create***(false)

Create and compile regular expression for extracting words from the input (“[a-zA-Z0-9_\-\']+”)

WHILE input has a matching **word** do:

    # Check to see if this word has been added to the Bloom filter

    If ***bf_probe***(**bf**, **word**) == FALSE then continue loop

    If hash table contains word and word does not have newspeak then:

        Insert this **word** to **bad_speak_list**

    If hash table contains word and word does have newspeak then:

        Insert this **word** to **new_speak_list** with newspeak translation

Print result

If **bad_speak_list** is not empty and **new_speak_list** is not empty then:

    Print “Thoughtcrime” message

Print **bad_speak_list**

Print **new_speak_list**

Else If **bad_speak_list** is not then:

Print “Thoughtcrime” message

Print **bad_speak_list**

Else If **new_speak_list** is not empty then:

Print “Rightspeak” message

Print **new_speak_list**

Delete **ht**

Delete **bf**

Delete **bad_speak_list**

Delete **new_speak_list**

Close files