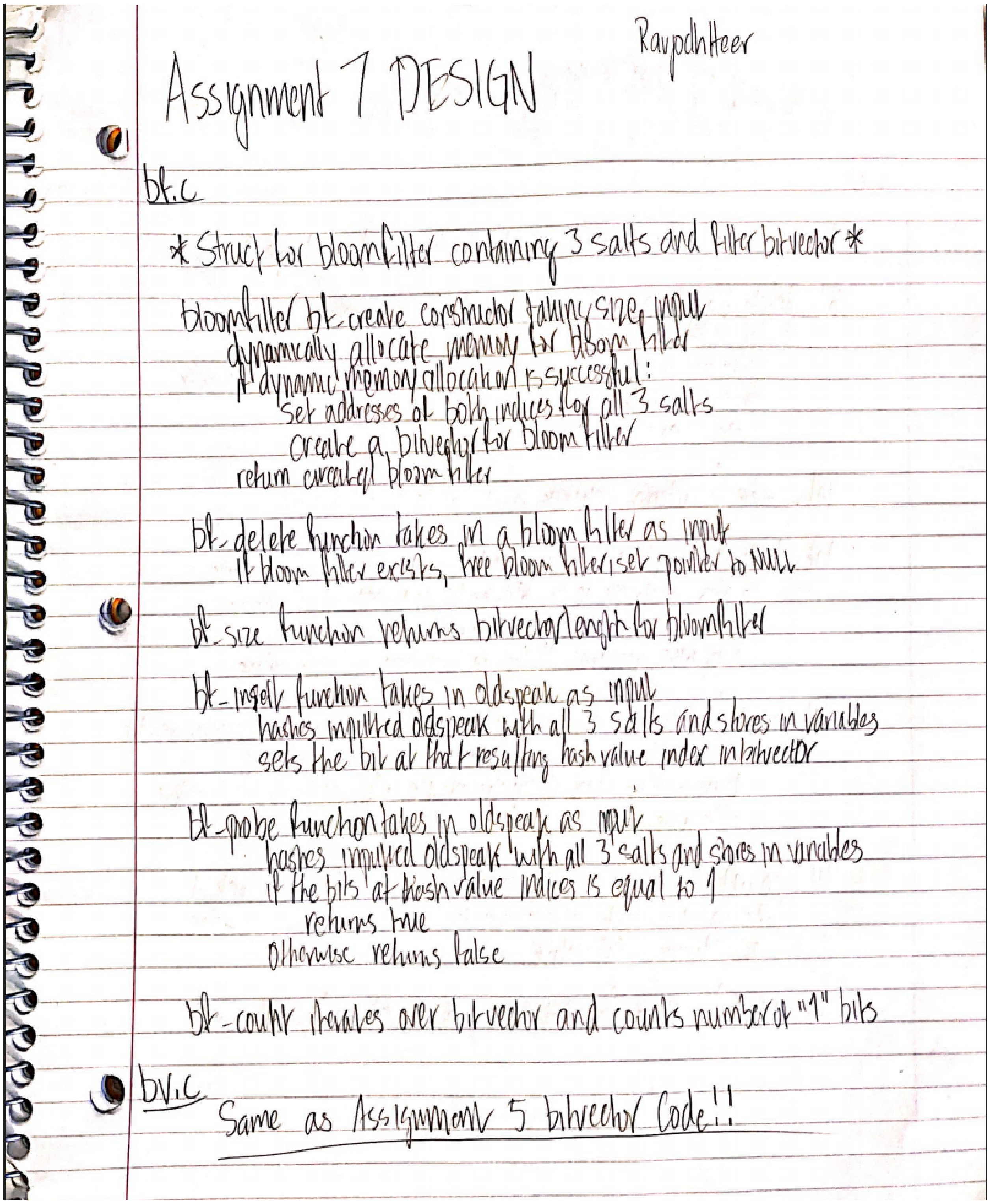
Ravjodh Heer

Assignment 7: The Great Firewall of Santa Cruz - DESIGN.pdf

Purpose:

_____The purpose of this lab is to use bloom filters, hash tables of linked lists containing nodes to create a firewall for inputted text. This 1984 inspired assignment involves censorship of certain banned words. The program we develop will aid in catching unwanted words found in inputted text. Those unwanted words will then be evaluated using bloom filters and a hash table of linked lists containing nodes in order to confirm if the current word is completely unwanted (a bad word) or if it is simply an utterance of oldspeak, to be replaced with a newspeak alternative. The program will scan an inputted text and return a government issued message regarding the results and will return the bad words uttered in the text, as well as the oldspeak uttered in the text with its corresponding newspeak alternative. If the inputted text contains bad words the citizen will be punished, however, if the inputted text contains oldspeak, the citizen will undergo a refinement program in which they will be taught newspeak. In cases where both are present, the citizen must perish through both means of punishment.

Pseudocode Draft:



Raypdhtteer

Assignment 7 DESIGN

bf.c

```
* Struct for bloomfilter containing 3 salts and filter bitvector *

bloomfilter bf_create constructor taking size input
    dynamically allocate memory for bloom filter
    if dynamic memory allocation is successful:
        set addresses of both indices for all 3 salts
        create a bitvector for bloom filter
    return created bloom filter

bf_delete function takes in a bloom filter as input
    if bloom filter exists, free bloom filter, set pointer to NULL

bf_size function returns bitvector length for bloomfilter

bf_insert function takes in oldspeak as input
    hashes inputted oldspeak with all 3 salts and stores in variables
    sets the bit at that resulting hash value index in bitvector

bf_probe function takes in oldspeak as input
    hashes inputted oldspeak with all 3 salts and stores in variables
    if the bits at hash value indices is equal to 1
        returns true
    otherwise returns false

bf_count iterates over bitvector and counts number of "1" bits
```

bv.c

Same as Assignment 5 bitvector Code!!

Assignment 7 DESIGN

ht.c:

* Hashtable Struct and Constructor provided in Asgn Doc *

```
ht_delete function takes in hashtable as input
  if the hash table exists
      iterates over size of hash table and calls linkedlist delete
      frees hashtable pointer
      sets pointer to null.

ht_size function returns hash table size pointer

ht_lookup function returns node with inputted oldspeak
    stores hash value of oldspeak and salt into variable
    if hash value index is empty, returns NULL
    otherwise, returns ll_lookup function of hash value

ht_insert function takes in oldspeak and newspeak and inserts into ht
      stores hash value of oldspeak and salt into variable
      if hash value index is empty, creates a linked list at hash index
      otherwise, calls ll_insert function to insert values

ht_count function counts the number of NULL linked
    lists in the hash table subtracts from size of hash table
    and returns that value.

ht_print function for debugging purposes, not important!
```

Assignment 7 DESIGN

ll.c:

* linked list struct and constructor provided in assignment doc *

```
ll-delete function takes in linked list as input
    if the linked list exists
        iterates through all the nodes in the linked list
            calls node-delete on each node
        frees linked list pointer
        sets pointer to NULL

ll-length function returns the length of the linked list

ll-lookup function takes in oldspeak as input
    if mtf is true
        exchanges pointers of current node to the head
        exchanges head pointers and tail pointers to
        accomodate for moving to front
    otherwise returns oldspeak

ll-insert function takes in oldspeak and newspeak
    creates a new node of oldspeak and newspeak
    exchanges head pointer to new node pointers
    exchanges tail pointers to new node pointers
    adds to the length of the linked list

ll-print function for debugging purposes, not important!
```

node.c

* Same as Assignment 6 node ADT *

<u>Pseudocode Final:</u>

<u>bv.c:</u>

```
BitVector bv_create function with length as input parameter
    Allocates memory for BitVector
    if memory allocation fails, return NULL
    set vector length pointer to length
    equation to calculate byte allocation for  vector allocation
    allocates memory for v->vector
    if memory allocation fails, free v and return NULL
    returns vector v


bv_delete function deletes the bitvector
    frees the pointer to v->vector and frees v and sets v to NULL


bv_length returns length pointer of v


bv_set_bit function takes in vector and index i as input
    creates a value to store result of masking index value by 1
    i/8 index of vector is OR'd with mask value at index, OR preserves the value (sets bits
      to 1)


bv_clr_bit function takes in vector and index i as input
    creates a value to store the result of inverting the mask of index i
    i/8 index of vector is AND'ed with mask value, AND sets previous values to 0 (clears
    bits to 0)

bv_get_bit function takes in vector and index i as input
    modulus value storing index position is created
    mask value of 0x1 is stored in mask variable
    i/8 index of vector is right shifted by modulus & mask

bv_print function for debugging reasons, not important
```

node.c:

```
stringdup function takes in a const char string as input
    returns strcpy of string with dynamically allocated space for new string copy

node_create function takes in symbol and frequency as input
    dynamically allocates memory for Node
    if dynamic memory allocation is successful
        if oldspeak is null, sets node's oldspeak to null
        if oldspeak isn't null, sets node's oldspeak to stringdup function of oldspeak
        if newspeak is null, sets node's newspeak to null
        if newspeak isn't null, sets node's newspeak to stringdup function of newspeak
        node's next child pointer set to NULL
        node's prev child pointer set to NULL
    returns created node
```

```
node_delete takes in a node and frees all memory
  if the node exists
    frees oldspeak
    frees newspeak
    frees the node and sets pointer to NULL
    sets node pointer to null
```

node_print function for debugging purposes, given in assignment doc

<u>bf.c:</u>

Bloom Filter constructor given in assignment document with provided salt values and bf->filter bitvector

bf delete function takes in bloom filter, calls bitvector delete on bf filter bitvector and sets pointer to null

bf size function returns bitvector length given bloomfilter

```
bf insert takes in bloomfilter and oldspeak and inserts oldspeak's hash values at respective indices
  creates variables to store the 3 hash values of hashing inputted oldspeak
  mods returned hash values by bloomfilter size
  sets the bits at each of those indices in the bloomfilter's bitvector
```

```
bf probe checks to see if a given oldspeak hash value has indices already set in the
 bitvector
    creates variables to store the 3 hash value of hashing inputted oldspeak
    mods returned hash values by bloomfilter size
    if the bits at the 3 hash value indices are set to 1, returns true
    otherwise returns false
```

```
bf count function counts the number of set bits in the bloom filter's bitvector
    creates counter variable, loops through the bitvector and increments counter every
     time a bit set to 1 is detected
```

bf print function calls bv print for bitvector

ll.c:

declares external variable for seeks

declares external variable for links

linkedlist struct given in assignment doc

```
linked list ll create function takes in mtf boolean
    dynamically allocates memory for linked list
    if memory allocation successful
        set length pointer to 0
        set mtf boolean pointer to inputted mtf boolean value
        create a NULL node at head pointer
        create a NULL node at tail pointer
        set head pointer's next pointer to tail pointer
        set tail pointer's prev pointer to head pointer
    return linkedlist

ll delete function takes in a linked list and frees all memory
    if the linked list exists
        loop through length of linkedlist + 2 for head and tail node and call node delete on
          each node
        free linked list pointer
        set pointer to null

ll length function returns length pointer

ll lookup function takes in oldspeak and searches for node
    increment seeks to signify function being called
    loops through each node between head and tail
        string compare node's oldspeak to inputted nodespeak
            if move to front is true then swaps pointers around in order to move the node to
              front
            returns the node after swapping
        increments links outside string compare
    if node not found, return null

ll insert function takes in oldspeak and newspeak and inserts created node into linked
  list
    calls ll lookup to check if oldspeak already exists in linked list
    creates a new node with given oldspeak and newspeak
    sets respective node pointers to head/tail and increments ll length pointer by 1

ll print function iterates through nodes between head and tail and calls node print
```

<u>ht.c:</u>

```
struct for hashtable given in assignment document

constructor for hash table given in assignment doc with ht size, ht mtf and ht lists

ht delete takes in a hash table and frees all memory
    if the hash table exists
        loop through the hash table and delete each non-NULL linked list using ll delete
        free ht lists pointer
        free ht pointer
        set pointer to null
```

```
ht size function returns hash table size pointer

ht lookup function takes in oldspeak and searches hash table for given oldspeak
    creates an index variable to store hashed salt value
    if hashtable at stored index doesn't exist, return null
    otherwise call ll lookup on hashtable index and given oldspeak

ht insert function takes in oldspeak and newspeak and inserts into hashtable
    creates index variable to store hashed salt value
    if hashtable at stored index doesn't exist
        call ll create to create a linked list at the index
    call ll insert to insert given oldspeak and newspeak at hash table index

ht count returns the number of non-NULL linked lists in the hash table
    creates counter variable and loops through size of hashtable and increments counter
     when a null list is found
        returns hashtable size minus null counter

ht print function iterates through ht size, creates a linked list at index and prints NULL
or calls ll print based on index value
```

banhammer.c

regular expression implementation given by Eugene in 6/1/21 lab section

```
declare seeks and links variables for variable tracking

lowercase converter converts given string to lowercase

main function that takes in getopt arguments
    default hashtable size set to 10000
    default bloomfilter size set to 2^20 (1048576)
    mtf boolean set to false as default
    statistics boolean to enable printing of statistics (default false)
    sets opt to 0 for getopt function
    getopt function while statement
        switch statement for getopt function
        case h defined as help statement
            Prints help statement
        case t for hashtable size
            Changes hashtable size to optarg
        case f for bloomfilter size
            Changes bloomfilter size to optarg
        case m for move-to-front rule
            sets move to front boolean to true
        case s for printing stats to stderr
            Sets statistics boolean to true
        default case printing out error statement if user input is invalid
            prints default case statement
    create bloom filter and hashtable
    regex creation taken from assignment doc example of parsing module
    read in badspeak file
    while loop with fscanf to scan in each bad word and add to bloomfilter and hashtable
    read in newspeak file
    while loop with fscanf to scan in each new word and add to bloomfilter and hashtable

    create a linked list for badspeak
    create a linked list for newspeak

    fscanf input
        change word to lowercase
        check bloom filter for words
        check hashtable
        if word in hashtable
            record words

    for recorded words
    print respective message
    print words
    if statistics isn't set to true, print out messages if badspeak or newspeak linked lists
```

```
  contain any values
print statistics to stdout


free and close files, delete ADTs, and clear regex then return 0 and end program
```