**HOMEWORK 5: HASH TABLE OF STRINGS**

For this assignment, you will work with a hash table (with chaining) of strings. All code should be written in C.

1. **(5 points)**Define a struct node for a linked list node whose information includes a single string and an unsigned integer count.
2. **(5 points)**Define a struct hashtable that includes a pointer to a dynamically-allocated (from the heap) array of pointers to struct node, and the size of the array (as an unsigned integer).
3. **(5 points)**Write a function that initializes a hash table. The prototype should be:

void init\_table(struct hashtable\* T, unsigned size);

On input, T points to an existing but uninitialized hashtable struct. On output, the struct should be initialized to an empty (all zero pointers) hash table of the specified size.

1. **(25 points)**Write a function that adds a string to a hash table. The prototype should be:

void add\_to\_table(struct hashtable\* T, char\* str);

This function should

* 1. Compute a hash h for the string, using the value of Dan Bernetein's **djb2** hash function for strings (given below) **modulo the size of the hash table**.

unsigned long

hash(char \*str)

{

unsigned long hash = 5381;

int c;

while (c = \*str++)

hash = ((hash << 5) + hash) + c;

return hash;

}

* 1. Search the linked list pointed at element h of the hash table, for a node containing a string equal to str.
  2. If such a node n is found, then increment the counter in node n.
  3. If no such node is found, then create a new node whose string is equal to str and initialize its counter to 1. Add this node to the front of the linked list pointed at element h.

It is up to you if function add\_to\_table makes a copy of the string or not, but you will need to use the function correctly in the main program (see below) and make sure there are no memory leaks.

1. **(10 points)**Write a function that displays (to standard output) all nodes in a hash table. The prototype should be:

void show\_table(struct hashtable\* T);

Specifically, for each list in hash table T (in order), for each element in the list, write the count and the string, on its own line, in the format

count str

1. **(10 points)**Using the above functions and any other helper functions you need, write a main() program that reads from standard input: first, an integer (on its own line) specifying the size of hash table to create; then, one word per line (with no spaces anywhere). Each word should be converted to lower case (hint: see function tolower), and added to the hash table. When the input has been consumed, display the hash table (counts and words) to standard output. You may assume that words are 32 characters long or shorter; you may exit cleanly with an appropriate error message if the input file contains any words that are too long. Example input files and their expected outputs will be posted to the git repository for class. As a short example, the input

29

This

is

a

test

file

and

this

is

a

short

example

should have corresponding output

1 example

2 this

2 is

1 and

2 a

1 short

1 file

1 test

Collect all this together into a single, well-documented file named count.c, and upload it to Canvas. Your code will be compiled and tested on pyrite.