Thursday Problem: Hash Table
Due 4/6/2012 over Webcourses

Problem

You are compiling a list of famous people and so far you have ***936***. You want to keep adding to your list but would like a fast insert, delete and search time. So you decide to store your famous people in a hash table. To handle collisions your hash table will use the ***quadratic probing*** method.

After your program has read in the list of famous people into the hash table, you will present the user with statistics about the hash table. (These are specified below.) Then, your program will prompt the user for a person to search. You will then determine whether or not the person entered by the user is in the hash table or not and print out a message to that effect.

You should allow the user to continue checking people until he/she wants to quit. Sample output showing roughly how your program should run is included below (Program output in italics, User input in normal font):

Welcome to the Famous People List!
Enter the name of your people file.
FamousPeople.txt

Great, your list has been stored in memory.
There are 8012 empty entries.
The maximum number of collisions was 21.
The total number of collisions was 924.
The average number of collisions was 1.0.

Would you like to check another famous person(yes=1,no=0)?
1
What person would you like to search for?
Vincent_Van_Gogh
Vincent_Van_Gogh is a valid famous person, found in entry _____.

Would you like to check another word(yes=1,no=0)?
1
What person would you like to search for?
Sigmund_Freudian
Sigmund_Freudian is not a valid famous person.

Would you like to check another word(yes=1,no=0)?
0
Thank you for using the Famous People List!

Note: This sample is NOT accurate. It is just included to indicate the desired FORMAT for the I/O.

Hash Function:
Here is the hash function I would like you to implement:

Let the famous person to be hashed be $w_1w_2...w_n$ where each w_i $1 \le i \le n$, is a single character. Also, let val(x) be the numerical equivalent in between 0 to 25 of the letter x. In particular, A = 0, B = 1, ... , Z = 25. Assume that all people are formed with lower or upper case letters only, thus ignore all underscores "_".

$f(w_1w_2...w_n) = [val(w_1)*26^{n-1} + val(w_2)*26^{n-2} +...+ val(w_{n-1})*26^1+val(w_n)*26^0] \bmod 1867.$

As the function indicates, your array size for your hash table should also be 9109. Here is one small example:

f("dog")=(val('d')*26^2+val('o')*26^1+val('g')*26^0)%9109=(3x26^2+14x26+6)% 1867=_____.

So, dog would be stored in the linked list with index _____.

Constraints:
Table Size: 1867
Max word size: 100

For curiosity
Experiment with different hash table sizes as well as prime/not prime hash table sizes. What do you observe?