

This is an Extra-Credit assignment. Completing any part of this can only improve your course programming %. You can earn up to 20 points toward your overall programming score; however your programming % may not exceed 100%. You may work on this in groups of two if you wish; each group member will receive half of the credit awarded. All group member names should appear in all files.

To get credit for this assignment, create a subdirectory named prog6 and do all of your work for this assignment in that directory. The required files will be electronically collected at the specified due date/time. Absence of this directory and related files will mean you elected to not attempt the extra credit.

Objectives.

- To understand the hash table implementation provided in hash.cpp, in particular:
 - use of C++ templates
 - use of function pointers
 - use of STL templated containers
- To implement and study the effectiveness of various hash functions

Procedure

First copy the files hash.cpp, hsamp.cpp and dict.txt from the class library.

Part I. Hashing with linear probing. 10 points.

Here you will consider the impact of hash function quality & load factor in a table using linear probing by running an experiment 100 times.

1. Create two hashing functions.

One should be a variation on the mid-square concept, the other of your own choice. Feel free to borrow from hash examples on the web, but give credit (URL, etc) to the reference.

Code your function using the signature provided in the example. Place them in the driver program in the location specified. Include a short comment block which describes the hash method.

2. Amend the driver program to test both your hash functions. The statistics of interest are:
 - average maximum # of probes required for the 100 trials
 - average # of probes required for 100 trials

Your driver program should compute these numbers for both hash functions with Load Factors: 0.5 and 0.7 (There will be 4 sets of outputs). For your table size, select a prime number > 100.

Your program should print these values suitably annotated.

Your program should be sufficiently documented to convince the reader that the methodology is correct.

BONUS: 5 point bonus if one of your hash functions is so good that the average maximum # of probes is 5 or less for load factor 0.7. Your program output should make it evident that you deserve the bonus.

Part I. Turnin:

name your driver program hashp1.cpp

Create a photo file named hashp1.photo containing:

```
g++ hashp1.cpp
a.out
```

Part II. Hashing with chaining. 10 points.

The hash table class provided is 'set up' to implement chaining. That is: the element of each table element are list<T>.

In this exercise you will first modify the insert function so that it implements the chaining protocol. You will then implement a corresponding find function. Normally the find function would return the matching record; but here you will return the number of probes to the list required.

Modify the driver program to test your implementation by creating a hash table as in Part I. Insert a random set of keys so that the load factor is 3. Then invoke your find function using each key; determine and print the average number of probes required.

Repeat the above with the second hash function.

To receive significant credit, your code MUST be reasonably documented and your output suitably labeled.

Rename hash.cpp as hashchain.cpp

Name your driver program hashp2.cpp (don't forget to change the #include to hashchain.cpp)

Create a photo file named hashp2.photo containing.

```
g++ hashp2.cpp
cat hashchain.cpp
cat hashp2.cpp
a.out
```