1. Study the Chapter 9 slides on the hash tables.

2. Implement the following functions in the HashData project.

   ```
   insert()
   contains()
   toStream()
   bernsteinHash()
   knuthHash()
   ```

   The specification for the hash table is in HashData.hpp, and the hash functions are in HashFunctions.hpp.

   Hand in

   HashData.hpp
   HashFunctions.hpp

   electronically per the instructions for your course.

   This project uses the ListL data structure for the chains. Use the prepend() list method to implement the insert() hash table method, and the contains() list method to implement the contains() hash table method. Each chain is a linked list of CAMetrics with template parameter string. So, the project also depends on the CAMetrics project for the purpose of collecting performance statistics on the hash table.

   The main program prompts the user for the hash function type (Knuth or Bernstein), the size of the hash table, the number of keys to load, and the file of keys to install into the hash table. The file words.txt contains a list of words to use as keys in the hash table for testing your code. The program loads the table with the keys from the file in order starting from the beginning. Then, it goes back to the beginning of the file and tests twice as many keys as were loaded, taking them in order from the beginning of the file. For example, if you specify 100 keys to load it loads the first 100 keys from the file into the table then goes back to the beginning of the file and tests the first 200 keys. If your code is correct, the program should find the first 100 keys and should declare that the next 100 keys are not in the table.

   There are four unit tests in the dp4ds software distribution, two for the Knuth hash function and two for the Bernstein function. One of the unit tests for each function is a small example with only seven slots and ten symbols and includes a dump of the hash table. If your implementation matches that hash table dump it will probably pass the larger test.