CIS 291 – Data Structures in C++ - Spring 2005
Homework #4
HW #4 due: Thursday, February 17th, BEGINNING of class

Purpose:
Practice with hashing (using linear probing)

How this will be turned in:
Use **~st10/291submit**, called from the directory on cs-server where the files you wish to submit are stored.

Homework Problems:
Consider the **htable** pseudo-UML posted along with this assignment.

Consider also the posted file **htable.h**, an example of a header file for a class implementing this particular pseudo-UML using a static-array-based hash table using linear probing.

Finally, consider the posted file **test_htable.cpp,** a testing main that partially tests the class htable.

1. **Carefully** read the htable pseuod-UML and **htable.h**. Note that there are several places where you will need to modify **htable.h**. When you do so, add your name in the opening comment block, and change the date last modified.

2. Write a file **htable.cpp** implementing **htable.h**.

   You may use whatever at-least-semi-reasonable hash function logic you wish, as long as it includes modulo arithmetic as a "final" step (your result needs to be modulo HTABLE_CAP...!) Note that some examples in **htable.h**'s comments need to reflect your hash function's expected behavior.

   Note the miscellaneous notes below, also.

3. Run **test_htable.cpp** for your implementation, and debug and repeat until you believe all of those tests have been passed. You are welcome to add additional tests to **test_htable.cpp** if you wish --- you may NOT remove any of the tests there. IF you add anything to **test_htable.cpp**, then add your name in its opening comment block, and change the date last modified.

4. Redirect **test_htable**'s output to a file, **hw04_output**, when you are happy with it.

When you are done, submit the following files:

htable.h (because you should have modified it), **htable.cpp**, **hw04_output**
and, IF you added to it, **test_htable.cpp**.

Miscellaneous notes:
* note: htable is **not** a template class (for this assignment, anyway... 8-))
* you MUST use hashing appropriately within this class; it must be used for storing and searching/removing.
* you MUST use linear probing;
* note the static constants for **-1** for NEVER_USED, and **-2** for PREVIOUSLY_USED. You are required to use these appropriately within htable's implementation...

* where is the **hash** function in the pseudo-UML? It deliberately isn't there --- that's a **private** method within the **htable** class... [an implementation detail]
* hint: remember to add modulo the table size WHENEVER you increment a table index...!
* why the private method **get_index**? Well, it isn't the public user's business how this table is implemented (as an array, a linked list, buckets and chaining, etc.). But, privately, there are times within the implementation where being able to call and get the array index where a target actually is would be convenient...
* having trouble debugging your implementation? Add debugging cout's to htable.cpp, **temporarily**...