CISP 430 Data Structures
Project 6: Hash Table

MUST BE TURNED IN BY Thursday DEC. 12 – late work will not be accepted

For this assignment you will implement ADT hash table as a C++ class: HashTable with the following functionality:

```
//Post: An empty HashTable is created and initialized
HashTable( );

//Post: The HashTable is destroyed and all memory is released
~HashTable( );

//Pre: There is room in the HashTable
//Post: The ItemType is inserted into the HashTable
//according to its key
void Insert ( const ItemType& );

//Pre: The HashTable is not empty
//Post: The ItemType with this key is returned.
//If the table does not contain an ItemType with this key it
//returns ItemType ("KEY NOT FOUND","KEY NOT FOUND")
ItemType Retrieve ( const KeyType& key );

//Pre: The HashTable is not empty
//Post: The ItemType with this key is removed from the
HashTable.
//If the key is not found, it does nothing
void Delete ( const KeyType& key );

//Post: Returns true if there is nothing in the HashTable
bool IsEmpty ( ) const;
```

HashTable will have a fixed sized array of linked lists using your **List** class with `ItemType` `typedef`ed to be an object of type Record using my Record class. You will add a method to `List`:

```
//Post: If the list was empty then ITEM is the only item in the
//list. Otherwise, ITEM is inserted into the list so that the
//list is ordered from first key to last key.
void OrderedInsert ( /*in*/ const ItemType& ITEM );
```

You will use `OrderedInsert` to insert items into the `HashTable`. This will be used to improve retrieval times, since you do not need to search the entire list to see if an item is in the

`HashTable.`
You will read in my file of Records and insert them into your HashTable.

Once you have finished your `HashTable`, perform the following tests:

Adjust the table size.
Change your hash function.
Use `Insert` rather than `OrderedInsert`

Try to determine the best fit of hash function, table size and insertion method.

Then run a series of tests that pit your HashTable against your BSTree, timing random retrievals and deletions from each. Which performs best? Write a report of your findings. It may be in tabular form. This report must be a professionally formatted file; clear and easy to read and understand.
You will turn in:

HashTable.h
HashTable.cpp
TestResults.doc

Your HashTable must compile and run with my main, Record, List, and STRING. If it does not, you will receive zero credit for this assignment. Be sure to follow the specifications exactly. This includes the names of the files.

Submit your files to the D2L drop box before it closes.

DO NOT ZIP THE FILES