CMPSCI 182, Data Structures and Program Design

Project 6 – Hash Tables (30 points)

NOT ASSIGNED

For this project, you will take the PhoneBook class you wrote for Project #5 and re-write it so that it contains a hash table as a data field. In other words, instead of storing Person objects in a binary search tree, the PhoneBook class will store these Person objects in a hash table.

This new application should still provide the same operations as it did in Project 5, but will perform these operations on a hash table instead. The key to this project is in developing a good hashCode() function, which translates a key value into an index number.

If this was not done in Project 5, you should organize your project into the following classes:

Person, which represents that name and phone number of a person, you will store instances of this class in the phone book.

PhoneBook, which represents the phone book. The class should contain a hash table as a data field. This table contains the people in the book. The phone book should contain the add, delete, find, change, quit, save and restore methods.

Menu, which provides the program's user interface—contains a main() method which creates a PhoneBook object, displays the PhoneBook's methods as different menu selections and invokes the PhoneBook method the user selects.

HashTable, which is the ADT HashTable. This is the class which contains the PhoneBook's collection of data (all of the People objects in the PhoneBook), as well as the operations which can be performed on that collection of data.

The key to this project is in the implementation of a hashCode() method, which "translates" a key value into a numeric index value, which identifies the location in which an item will be "stored" in the hash table.

You must also consider how to resolve a "collision". A collision is when two key values are translated into the same index number.