CSCI 335
Fourth assignment
Total: 100 points
Due 4/27/2015
Please, follow the blackboard instructions on writing and submitting programming assignments
We will not debug your assignment. It should run correctly to receive credit

Priority queues (100 points)

You are asked to implement a binomial queue (**bq**) linked with a hash-table. You can use code provided in the textbook. The **bq** is used for fast access of the minimum element and the hash-table for fast access within the elements in the **bq**.

(1) Implement the binomial queue. For testing use the strings in the document file used in the previous assignment. These would be the **keys** to be inserted into the priority queue. Implement a hash-table of the keys that are already in the priority queue. So, for any pair **<key,p>**, where **p** is the pointer of the the node that holds key in the priority queue, hash on the **key**, and store the pair **<key,p>** in your hash table. Also note that for every insertion, deleteMin, or merge the hash-table needs to be updated as well. As part of this implementation you have to create a private function
<Pointer to bq node> find(<Type of Key> **key**).
This function will return the pointer **p** that points to the node of **bq** that holds the **key**, or nullptr if **key** is not in the **bq**.

(2) Count and print out the total number of comparisons and assignments executed for the insertions of all the N elements into the binomial queue.

(3) Test the deleteMin() operation by applying a sequence of 10 deleteMin() and by printing out the result.

(4) Test the function find() as follows: Prompt the user to input a string **key**. Execute the private function find(**key**). If find returns a pointer to a node that indeed holds key printout that find() was successful. Otherwise, printout that find() did not find the **key**.

(5) You are ready to implement now the remove(**key**) operation. For a given **key**, find its position **p** (**p** is a pointer) in the priority queue using the hash table. If the key is found delete it from the **bq** (Hint: your code should percolate the key up all the way to the root and then delete the root). Test your implementation by applying a sequence of remove(**key**) operations, and verify their correctness. For instance after remove(**key**), find(**key**) should return "not found". Prompt the user 5 times to input a **key.** Execute then the remove(key) operation and verify (by printing whether the removal was successful or not).

(6) Write a faster insert(key) function for the binomial queue. In order to achieve that you have to modify the merge() function and make it specific to the merging of one element only.