### COSC 310 – Data Structures and Algorithms

### Assignment 8 – Hashtables and Sets

**Due:** November 21 at 2:00 AM - 40 points.

The objectives of this assignment are to:

1. Gain further experience with using interfaces.
2. Gain experience with implementations of a Hashtable.
3. Gain experience using Sets.
4. Gain further experience with dictionary applications.
5. Continue to practice good programming techniques.

### Task:

**Step 1:**

Provide a hash table implementation of a Dictionary (use the same interface as assignment 7). You may chose linear probe (open addressing) or separate chaining to handle collisions. You may use the hashCode() method built into Object and overridden in String to obtain your initial hash index. You still need to modify the return from hashCode() to accommodate the size of the array in your hashtable. **You do not need to handle rehashing the table to handle the case where you run out of capacity**. Instead, choose an initial size of sufficient size. **Note:** The order returned by the iterator will no longer be sorted.

**Step 2:**

Create a jUnit test program to thoroughly test your hashtable implementation of a Dictionary (**make sure you test various test cases with delete**).

**Step 3:**

Create a word index program (not in edu.iup.cosc310.util). The word index program must open a text file passed to it on the command line. Your program must process the text file identifying the set of unique words in the file and the page numbers on which each word occurs. Separate words on spaces, commas, tabs, newlines, carriage returns, dash, slash, periods, semicolons, colons, quotes, parenthesis, and square brackets (i.e. split on “\\W+”. A page is considered to be 50 lines and starts at page 1.

After processing all the words in the input text file, your word index program must print a list of all the found words, **in sorted order**, together with the list of page numbers where the words are found. The page numbers must be in sorted order and each page number only is printed once for a given word. Thus, **you must use a Set** for maintaining the list of page numbers. You can use one of Java’s Set implementations. The output is to be is sorted word order. Thus when iterating over the keys, **you will need to place the keys into List and then use Collections.sort to sort** them.

### Deliverables:

1. A. zip file containing the sources and class files of your interface, hash table implementation, and word index program. The .zip file must be named [Your name]Asmt8.zip.
2. File containing the captured output from running your test program.
3. File containing the captured output from executing your word count program against sylabus.txt of assignment 7.

### Bonus – 10 points:

1. Handle rehashing the table when the load factor reaches 50%. Have the initial size for the table be 11. Effectively double the size of the table and round up to the nearest prime number (you can hard code into an array the prime numbers that apply. Make sure to test the code in Junit tests.