CSE 331 - Project 4
Hash Sets

TA: James Daly
dalyjame@msu.edu
Due Date: 11:59 pm Oct 31, 2014

1 Project Description

In this project, you will complete an implementation of a hash set. You are given skeleton code (i.e., Main.cpp) that performs reading inputs and printing outputs as well as a class declaration for a hash set (HashSet.h) and signatures for three hash functions (HashFunc.h). Your job is to complete the implementation of the following methods in the `HashSet` class:

1. `NumBuckets`: The number of hash buckets used by the set
2. `Contains`: Checks whether an item is in the set
3. `Insert`: Adds an item to the set (if it was not already there). Returns whether the addition was successful.
4. `Remove`: Removes an item from the set. Returns whether the removal was successful.
5. `Clear`: Removes all items from the set.
6. `ForEach`: Invokes a function on each element of the set. The ordering is undefined, but it must be invoked precisely once on each item.
7. `MaxBucketSize`: The number of items in the most full bucket.
8. `PercentEmptyBuckets`: The percentage of buckets that contain no items.

You may also need to make adjustments to the constructor and destructor.

Additionally, you will be required to implement three different hash functions of your choosing in HashFunc.cpp. The hash functions must have significant differences. You should cite where each hash function comes from. Some options appear on page 195 of your textbook. Other options include the Java String hashCode, FNV hash, Jenkins hash, CRC32, MurmurHash, or Pearson hashing.

Some other methods have been provided for you but may depend on these methods to function. You may add additional methods to the `HashSet` class (such as for rehashing the entire set) as desired. Note that the `HashSet` class

is templated; any other methods you add should also be templated. You may assume that any type given will supply `operator==`. You **may** use the `vector` and `list` classes from the STL for this project.

Each `HashSet` has a designated load factor (the mean number of items per bucket). If the load factor is exceeded, you must increase the number of buckets and rehash the set. You are not required to rehash after removals although you are allowed to. The exact number of buckets is your choice.

You must maintain the set invariants; an item may only be in the set once. You must also handle requests to remove elements not in the set.

`Contains` and `NumBuckets` should have an expected runtime of $O(1)$. `Insert` and `Remove` should have an expected amortized runtime of $O(1)$. `ForEach` and `Clear` may take up to $O(n+k)$ time. `MaxBucketSize` and `PercentEmptyBuckets` should run in $O(k)$ time.

You must provide your own code; you may not copy another's code. Remember to read the project guidelines before you start coding. Remember to update the size member variable when you insert or delete an item. Leaking memory or other related issues will negatively impact your score.

Not all of the above methods are tested by the supplied test harness in Main.cpp. It is your responsibility to ensure that all of the required methods work properly. **If any of the required methods do not compile, your project will not be graded.**

2 Input Test Cases

The program has two inputs. The first is a file containing strings separated from whitespace. Your program will read the strings from the file and store them inside of the set (the included main file will do this for you). The second is the load factor for the set.

Your program will then construct a hash table using each of the three hash functions. For each set, it will output the number of elements in the set, the number of buckets with zero entries, and the maximum number of entries in a bucket. The `Test` function in `Main.cpp` will publish these statistics. You will use this data to determine which hash function is best. You may also gather other information to aid in your decision.

3 Project Deliverables

The following files must be submitted via Handin no later than 11:59 pm Oct 31, 2014.

1. `HashSet.h` - contains your implementation of the `HashSet` class.
2. `HashFunc.cpp` - contains your implementation of three different hash functions.
3. project4.pdf - file containing your answers to written questions

4 Written Questions

1. For each of the provided files, list the number of distinct words.
2. For the midsummer.txt file, identify which of your hash functions is best and justify your answer.
3. For the lines.txt file, identify which of your hash functions is best and justify your answer.
4. For the plurals.txt file, identify which of your hash functions is best and justify your answer.