Computer Science II — CSci 1200
Lab 11
Hash Table Implementation

Introduction

The hash table implementation from Lecture 20 uses a vector of lists. During the process of automatically resizing the table, a significant amount of copying and re-allocation occurs, most of it hidden inside the functionality of the list and vector classes. An alternative technique is to use our own linked list implementation. That way we can ensure that minimal copying and no reallocation (other than resizing the `m_table` vector) occurs during `resize_table`. Lab 11 explores this implementation.

Download the files:

```
http://www.cs.rpi.edu/academics/courses/spring07/cs2/lab11/hash_set.h
http://www.cs.rpi.edu/academics/courses/spring07/cs2/lab11/test_hash_set.cpp
```

Then, turn off all network connections.

Examine the code in `hash_set.h` and `test_hash_set.cpp`. You will notice several changes over the version covered in lecture.

- A `HashNode` is a declared inside the `hash_set` class. This node is *doubly-linked*, meaning that both `prev` and `next` pointers are used. It also includes a caching of the computed hash function value so that this *need not be recomputed* during the `resize_table` function.
- The `iterator` class, also declared inside the `hash_set` class, now includes a `HashNode` pointer instead of a list iterator.
- `operator--` is not provided — mostly for simplicity, although the (seemingly) random ordering of keys implies that moving in both directions is not particularly necessary.

You should also examine the test main program.

Several functions in `hash_set.h` have not been implemented. These should be implemented as part of the checkpoints of this lab.

Checkpoints

1. Implement the `insert` function, ignoring, for now, the call to the `resize_table` function. This function should implement a linked-list

insertion (at the front of the list is fine) once the correct list has been found (using the hash function) and it has been determined that the key value is not already in the list. The function should store the hash value (not the index) computed by the hash function in the `HashNode`. The function should increment the size counter and it should return the appropriate iteriator/bool pair.

2. Implement and test the `begin()` function of the `hash_set` class and the `next()` function of the `iterator` class. Note that when the `hash_set` is empty, `begin()` should return the `end()` iterator, and when `next` runs out of values in the hash table it should return the equivalent of the `end()` iterator. Code in the main function should be uncommented to test this function.

3. Implement and test the `erase` and `resize_table` functions. The `resize_table` function should reuse the existing `HashNode` and must include **NO** `delete` and `new` operations. Code in the main function should be uncommented to test this function.