15-122: Principles of Imperative Computation, Fall 2014
Lab 8: Hash Tables

Tom Cortina (tcortina@cs) and Rob Simmons (rjsimmon@cs)

Monday, October 20, 2014

For this lab, you will show your TA your answers once you complete your activities. You are expected to complete the first exercise for full lab credit.

SETUP: Make a directory `lab8` in your private `15122` directory and copy the required lab files to your `lab8` directory:

```
cd private/15122
mkdir lab8
cp /afs/andrew.cmu.edu/usr9/tcortina/public/15122-f14/hset-*.c0 lab8
```

1 Removing Elements from Hash Tables

Recall that we implemented a hash table using separate chaining as an array of singly-linked lists with no dummy nodes. Review the code in `hset-lab.c0` to make sure you understand the representation for a hash table as an `hset` along with the functions `hset_insert` and `hset_lookup`.

Exercise 1. Complete the `hset_remove` function that has two parameters: the hset `H` and an element `x`. The function searches the appropriate chain for the key stored in element `x`. If the key is found, that element is removed from `H`.

The algorithm to perform this operation is similar to the lookup and insert algorithms. You need to determine which chain to examine first. Then traverse the chain, node by node, looking for a node that has an element that has a matching key. If you find a match, then remove the entire node from the list by linking the node before this node to the node that is after this node. BE CAREFUL: There is a special case here to consider. Do you see it? Draw pictures to help you visualize the procedure.

To test your solution, add code to the `main` function in the file `hset-testremove.c0`. The code we gave you creates and inserts 9 items to a basket. Your additional code should try to remove items from the basket based on the fruit (e.g. remove a pear, remove a banana, remove a melon, etc.). The color should not matter because of the way `elem_equal` and `elem_hash` were defined in `hset-client.c0`.

You should compile and run your code as follows:

```
cc0 -d hset-client.c0 hset-lab.c0 hset-testremove.c0
./a.out
```

Once you have your code completed and tested well, show your work (code and tests) to your TA for credit.

2 Resizing the Hash Table

Recall that load factor of a hash table is the number of elements stored divided by the capacity of the table (e.g. the number of chains). For example, a hash table with 60 elements and a capacity of 50 has a load factor of 1.2. (If elements are evenly distributed, then we'd expect 1.2 elements per chain.) In our `hset_insert` function, we did not take into account the load factor, so our chains can eventually get too long, reducing the effectiveness of a lookup operation.

Exercise 2. CHALLENGE Modify the `hset_insert` operation so that, before the element is added to the hash table, the function doubles the capacity of the hash table if the load factor is 1.0 or higher. Use a helper function to resize the table. Your solution should not create new nodes for the data elements. It should reuse the nodes already created. Just move the nodes to the new chain in the new hash table. (This is not as easy as it seems!)

You can test your solution by using the file `hset-testresize.c0`. You should compile and run your code as follows:

```
cc0 -d hset-client.c0 hset-lab.c0 hset-testresize.c0
./a.out
```