Lab 14

Comp 11 - Summer Session — Hashmap

14.1 Description

In this lab we are going to implement part of a hashmap. Our hashmap is limited to storing pairs with a key of string and a value of string for simplicity.

Our objectives are the following:

- Implement the hash function
- Implement the getValue function

14.2 Files

You may use the following code to help get you started. If you find it easier, you may (and are encouraged) to break this project into separate files, just remember to submit them all!

```
#ifndef COMP_HASH_MAP_H
#define COMP_HASH_MAP_H

#include <iostream>

struct node{
  node* next; // Points to the next element in bucket.
  // key and value are both string for this lab
  std::string key;
  std::string value;
  // Empty constructor
  node(){
    next = NULL;
  }

  // Abbreviated way to just assign parameters to
  // member variables. This can be done for constructors
  // and is known as an initializer list.
  node(std::string k, std::string v) : key(k), value(v){
    next = NULL;
  }
};


// Purpose: A Hashmap that consists of nodes
class CompHashMap{
// The buckets in a hashmap.
// An array of pointers is created
node** buckets;

unsigned int TABLE_SIZE; // How big our hashmap is

  // A private method that reallocates memory
  void resize();

  // A private allocate function that can be called
  // in each of the constructors
  void allocate(int size);

  // Computes index
  // Use length of screen and be sure to mod it by the TABLE_SIZE
  // so that the index generated falls within our table size
  //
  // You can play around with this function if you like
  // e.g. just return 0 to put everything in the first bucket
  // Note that we would have many different hash functions for
    different
  // data types.
  int computeHash(std::string key);
public:
  // Hashmap constructor
  CompHashMap(int size);

  ~CompHashMap();

  // Add a pair to the appropriate bucket
  void addPair(node* item);
```

```
57
58    // Prints out the Hashmap
59    void printHashMap();
60
61    // returns the value given a key
62    std::string getValue(std::string key);
63  };
64
65  #endif
```

Listing 14.1: The interface

```
1   #include "CompHashMap.h"
2   #include <iostream>
3
4
5   // A private method that reallocates memory
6   void CompHashMap::resize(){
7     // For now this is not implemented.
8     // We may want to have the ability to resize our table
9     // if there are too many collisions.
10  }
11
12  // A private allocate function that can be called
13  // in each of the constructors
14  void CompHashMap::allocate(int size){
15    buckets = new node*[size];
16    TABLE_SIZE = size;
17    for(int i =0; i < TABLE_SIZE; ++i){
18      buckets[i] = NULL;
19    }
20  }
21
22  // Computes index
23  int CompHashMap::computeHash(std::string key){
24    // TODO: Implement this
25      return 0;
26  }
27
28  // Hashmap constructor
29  CompHashMap::CompHashMap(int size){
30    allocate(size);
31  }
32
33  // Hashmap destructor
34  CompHashMap::~CompHashMap(){
35
36  }
37
38  // Add a pair to the appropriate bucket
39  void CompHashMap::addPair(node* item){
40    // Compute the hash to get an index
41    int index = computeHash(item->key);
42    // Point iterator the correct bucket
43    node* iter = buckets[index];
44    // Two cases, either nothing exists yet
45    // or something does and we need to iterate
46    // to the end of the chain.
```

```
47   if(iter==NULL){
48     buckets[index] = item;
49   }else{
50     // Traverse chain until we find
51     // the node that points to the end.
52     while(iter->next!=NULL){
53       iter=iter->next;
54     }
55     // Our end is now the new node
56     iter->next = item;
57   }
58 }
59
60 // Prints out the Hashmap
61 void CompHashMap::printHashMap(){
62   // Iterate through array
63   for(int i =0; i < TABLE_SIZE; ++i){
64     std::cout << "bucket " << i << " ";
65     // For each bucket in array, traverse linked list
66     node* iter = buckets[i];
67     while(iter != NULL){
68       std::cout << iter->key << "->";
69       iter = iter->next;
70     }
71     std::cout << "\n";
72   }
73 }
74
75
76 // returns the value given a key
77 // If the value is not found, then print an error message
78 std::string CompHashMap::getValue(std::string key){
79 // TODO: Implement this
80 //
81 //
82   return "No entry found";
83 }
```

Listing 14.2: The implementation

```
1  #include <iostream>
2  #include <map>
3  #include <string>
4
5  #include "CompHashMap.h"
6
7
8  int main(){
9
10   // Create our CompHashMap
11   CompHashMap myMap(10);
12
13   node* n1 = new node("ox","A male cattle");
14   myMap.addPair(n1);
15   node* n2 = new node("cow","A domesticated ungulate");
16   myMap.addPair(n2);
17   node* n3 = new node("pig","genus Sus");
18   myMap.addPair(n3);
```

```
19    node* n4 = new node("bat","webbed wing mammal");
20    myMap.addPair(n4);
21    node* n5 = new node("bear","carnivoran mammal");
22    myMap.addPair(n5);
23    node* n6 = new node("mouse","small rodent");
24    myMap.addPair(n6);
25    node* n7 = new node("camel","even-toed ungulate");
26    myMap.addPair(n7);
27    node* n8 = new node("jaguar","Panthera genus");
28    myMap.addPair(n8);
29    node* n9 = new node("cheetah","Acinonyx jubatus");
30    myMap.addPair(n9);
31
32    myMap.printHashMap();
33
34    std::cout << "myMap.getValue(\"ox\"):" << myMap.getValue("ox") <<
        "\n";
35    std::cout << "myMap.getValue(\"kangaroo\"):" << myMap.getValue("
        kangaroo") << "\n";
36    std::cout << "myMap.getValue(\"jaguar\"):" << myMap.getValue("
        jaguar") << "\n";
37
38
39    return 0;
40 }
```

Listing 14.3: main.cpp

14.3 Output

```
bucket 0
bucket 1
bucket 2 ox->
bucket 3 cow->pig->bat->
bucket 4 bear->
bucket 5 mouse->camel->
bucket 6 jaguar->
bucket 7 cheetah->
bucket 8
bucket 9
myMap.getValue("ox"):A male cattle
myMap.getValue("kangaroo"):No entry found
myMap.getValue("jaguar"):Panthera genus
```

Figure 14.1: The items in the hashmap

14.4 Refresher

Revisit the lecture slides for pictures of how hashmap works

14.5 Submission

```
provide comp11 lab14 your_cpp_and_h_files README
```

Listing 14.4: Submit Assignment

14.6 Going Further

Did you enjoy this lab? Want to try out some additional commands to go further?

- Try implementing the [] operator such that you can look up a value based on a key.