91.201 Computing III (Spring 2016) Instructor: Anna Rumshisky

Lab #5 – STL Types, Hash Functions

Handed out: 4/7/16	Due: 4/14/16

Problem

In this exercise, we will implement a hash table object using the implementation with a fixed-size array of linked lists. You can do this in stages:

STAGE ONE [20 pts total / items 1–5 : 15 pts, item 6 : 5 pts]

1. Rewrite the hash function below to take as an argument a value of STL type `string`, rather than `char *`:

```
unsigned long hash(unsigned char *str)  {
  unsigned long hash = 5381;
  int c;
  while (c = *str++) hash = ((hash << 5) + hash) + c;
  return hash;
}
```

2. Implement the version of a a hash table class that hashes STL strings to integers (i.e. keys are STL strings, values are integers).
3. Your class should provide a method to `insert` a value corresponding to a particular key, and a method to `get` the value that corresponds to a particular key.
4. Implement the version that does not allow collisions. If there is a collision, the insert method should return `false`.
5. Your implementation should use STL `vector` as an underlying container. You can either inherit from it or wrap it as a data member.
6. Overload subscript operator for your hashtable.

   Hint: this should be a trivial modification of your `get()` method.

   Hint: if you are inheriting from STL `vector`, invoking the subscript operator of the base class from inside derived class the can be done explicitly, for example:

```
return vector<int>::operator[](offset);
```

(OVER)

7. Test your implementation using this code:

```
#include <vector>
#include <string>
#include <iostream>
#include <stdexcept>
using namespace std;

int main() {
  HashTable h;
  string s1 = "John";
  string s2 = "Jake";
  string s3 = "Jane";
  h.insert(s1, 18);
  h.insert(s2, 21);
  h.insert(s3, 19);
  cout << "Jane: " << h.get(s3) << endl;

  string s4 = "Joanne";
  h[s4] = 20;
  cout << "Joanne: " << h.get(s4) << endl;
  cout << "Joanne: " << h[s4] << endl;

  int count = 0;
  for (auto item : h)
    cout << count++ << " " << item << endl;
}
```

STAGE TWO [20 pts]

8. Modify your class and its method to use a vector of STL `list` objects. Modify your member functions so that collisions would be resolved.
9. Test your implementation.

BONUS

10. Modify your class to take an arbitrary function that takes an STL sting object and returns an integer. Hint: this is a trivial modification!

Total: 40 pts (+ 5 pts bonus)