# COP 3530 Project 2, Part III

*Data Structures & Algorithms*
*Fall, 2014*

## Part III: Hash map with open addressing

Design, implement, and thoroughly test a hash map that supports *all three* of the open addressing techniques we discussed and handle multiple key types.

### Open addressing techniques

Either by implementing three different hash map variations, or by using generic programming techniques to define a configurable hash map, you are to support the *conceptual* function

```
hash( key, i ) = hash( key ) + i * probe( key )
```

where **i** is the probe attempt number (starting with 0) and the function **probe()** is *conceptually* defined, for each of the open addressing techniques as:

- **linear probing**:

```
probe( key ) = 1
```

- **quadratic probing**:

```
probe( key ) = i
```

- **rehashing**:

```
probe( key ) = hash2( key )
```

The reason the term *conceptually* has been emphasized, is because there is *no* requirement that the code be a *literal* implementation. For example, assume we're dealing with the rehashing variant: if one were to literally implement it, when inserting a particular key required 3 probe attempts, then the functions **hash()** and **hash2()** would each be computed three times, even though their values will always be the same for that key! Obviously, that is wasteful (and violates the programming principle that things that will have a constant value within a loop should be computed before the loop is entered).

## Hash functions

You will need to support parameterizing the hash map with a hash function appropriate for the key type, where the key type is one of:

- **`signed int`**
- **`double`**
- **`c-string`** — a **char** * pointing to a null terminated array of **char**s.
- **`std::string`**

### Equality functions

Don't forget to account for the fact that **operator==** isn't defined to compare the characters that comprise c-strings and std::strings!

## Map *key* & *value* types

### *Key* types

You will need to support parameterizing the hash map with the key type, where the key type is one of:

- **`signed int`**
- **`double`**
- **`c-string`** — a **char** * pointing to a null terminated array of **char**s.
- **`std::string`**

### *Value* types

You will need to support parameterizing the hash map with the value type, where the value type can be any type the user specifies.

## Revised and additional operations

### Revised hash map operations

The following functions shall now return an int value indicating the number of probe attempts required; this will facilitate the gathering of experimental data.

- **`int insert( key, value )`** — if there is space available, adds the specified key/value-pair to the hash map and returns the number of probes required, `P`; otherwise returns `-1 * P`. If an item already exists in the map with the same key, replace its value.
- **`int remove( key, &value)`** — if there is an item matching *key*, removes the key/value-pair from the map, stores it's value in *value*, and returns the number of probes required, `P`; otherwise returns `-1 * P`.
- **`int search( key, &value )`** — if there is an item matching *key*, stores it's value in *value*, and returns the number of probes required, `P`; otherwise returns `-1 * P`. Regardless, the item remains in the map.

## New hash map operations

These functions are solely for the purpose of gathering experimental data.

- **`??? cluster_distribution()`** — returns list of cluster sizes and instances, sorted by cluster size; for example, a map's whose backing array is populated thusly

  ```
  [ * | * | - | - | * | - | * | - | * ]
  ```

  (where * = occupied and - = unoccupied) would return the list ((1,2), (3,1)), since there is are two clusters of size one, and one cluster of size three (due to wrap-around). It is left to you to determine an appropriate return type.
- **`Key remove_random()`** — generate a random number, *R*, (1,size), and starting with slot zero in the backing array, find the *R*-th occupied slot; remove the item from that slot (adjusting subsequent items as necessary), and return its *key*.

This website is an original work, Copyright © 2014 by Dave Small. All rights reserved.