COP 3530 Project 2, Part II

Data Structures & Algorithms
Fall, 2014

Part II: Hash map with buckets

Design, implement, and thoroughly test an *bucket-based* hash map. At this time, the map need handle only *keys* of type `int` and *values* of type `char`.

Hash map operations

Hopefully, the function names are self explanatory, but just in case here are brief descriptions.

- **`bool insert( key, value )`** — if there is space available, adds the specified key/value-pair to the hash map and returns `true`; otherwise returns `false`. If an item already exists in the map with the same key, replace its value.
- **`bool 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 `true`; otherwise returns `false`.
- **`bool search( key, &value )`** — if there is an item matching *key*, stores it's value in *value*, and returns `true` (the item remains in the map); otherwise returns `false`.
- **`void clear()`** — removes all items from the map.
- **`is_empty()`** — returns `true` IFF the map contains no elements.

- **`std::size_t capacity()`** — returns the number of slots in the map.
- **`std::size_t size()`** — returns the number of items actually stored in the map.
- **`double load()`** — returns the map's load factor (~~size~~ occupied buckets = load * capacity).
- **`print( ostream& )`** — inserts into the ostream, the backing array's contents in sequential order. Empty slots shall be denoted by a hyphen, non-empty slots by that item's *key*. [This function will be used for debugging/monitoring].

Parameter value checking

For this part of the project, you may assume that all calls made to your hash map implementation will be made with valid arguments. However, things like the map being full, item not found, &c. should be handled as documented in the operation descriptions.