Hash maps

CS 261 Lab #9
Hash maps are **awesome**

With an **appropriate hash function** and **sufficient size**, they give us $O(1)$ inserts, removes, and searches!

And they’re **associative arrays**, letting us work with **key/value** pairs.
Hash maps are awesome true story

With an appropriate hash function and sufficient size, they give us $O(1)$ inserts, removes, and searches!

And they’re associative arrays, letting us work with key/value pairs
We’ll use the **buckets and chaining** approach to build hash maps

```
<table>
<thead>
<tr>
<th>0</th>
<th>key 1</th>
<th>value 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>key 2</td>
<td>value 2</td>
</tr>
<tr>
<td>3</td>
<td>key 3</td>
<td>value 3</td>
</tr>
<tr>
<td>4</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
```
We’ll use the **buckets and chaining** approach to build hash maps

```
"buckets"

0 -> key 1 | value 1
1 -> NULL
2 -> key 2 | value 2
3 -> key 3 | value 3
4 -> NULL

key 4 | value 4
```
We’ll use the **buckets and chaining** approach to build hash maps.

```
We'll use the buckets and chaining approach to build hash maps.
```

```
null
```

```
key 1 | value 1
```

```
key 2 | value 2
```

```
key 3 | value 3
```

```
key 4 | value 4
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```

```
NULL
```
To **insert** a key/value pair...
To *insert* a key/value pair...

1) **Compute the hash** for the key
To **insert** a key/value pair...

1) **Compute the hash** for the key

2) **Determine the bucket** in the hash map this key/value pair belongs in

\[(\text{hash} \mod \# \text{ of buckets})\]
To insert a key/value pair...

1) **Compute the hash** for the key

2) **Determine the bucket** in the hash map this key/value pair belongs in
   \[(hash \mod \# \text{ of buckets})\]

3) If the bucket doesn’t have any items in it, **make the bucket point to our item**. If it does have other items, **add our item to the chain**.
To **lookup** a value...
To *lookup* a value...

1) **Compute the hash** for the key
To lookup a value...

1) **Compute the hash** for the key

2) **Determine the bucket** in the hash map this key/value pair belongs in
   
   \[
   (\text{hash} \mod \# \text{ of buckets})
   \]
To lookup a value...

1) **Compute the hash** for the key

2) **Determine the bucket** in the hash map this key/value pair belongs in 
   \((\text{hash} \mod \# \text{ of buckets})\)

3) Examine each key/value pair in the bucket until you **find the matching key**, then return the key’s value.
Example: adding the key “hello” with a value of “world”
Example: adding the key “hello” with a value of “world”

Hash “hello” to a number (e.g., 581)
Example: adding the key “hello” with a value of “world”

Hash “hello” to a number (e.g., 581)

Mod that number by the number of buckets in our hash map (e.g., 581 % 5 = 1)
Example: adding the key “hello” with a value of “world”

- Hash “hello” to a number (e.g., 581)
- Mod that number by the number of buckets in our hash map (e.g., 581 % 5 = 1)
- Add the key “hello” with a value of “world” to bucket 1
Example: adding the key “hello” with a value of “world”

Hash “hello” to a number (e.g., 581)

Mod that number by the number of buckets in our hash map (e.g., 581 % 5 = 1)

Add the key “hello” with a value of “world” to bucket 1
In today’s lab you’ll implement the **insert** and **contains** methods for a hash map
In today’s lab you’ll implement the `insert` and `contains` methods for a hash map.

Then you’ll use the hash map to build a spell-checker.
In today’s lab you’ll implement the **insert** and **contains** methods for a hash map.

Then you’ll use the hash map to **build** a spell-checker.

This is a **very simplified** version of the hash map you’ll need to implement in Assignment #6 (but with a better hashing function!)
Your program will need to read the `dictionary.txt` file.

If you use the **Makefile** to compile this lab (via Xcode, Eclipse, or manually), place `dictionary.txt` in the same directory as the Makefile.

**Visual Studio** users will need to place `dictionary.txt` in Visual Studio’s build directory (usually `yourProjectDirectory\Debug`).
Download the files from
http://dropline.net/cs261/lab9

Implement the insertMap, containsKey, and tableLoad functions in hashmap.c

In main.c, use the hash map to check whether words the user enters are spelled correctly

Experiment with different values of size in main.c to see how it impacts the hash map’s speed and table load