## Goals

Implement a statically typed `hash_map`. See `hash_map.h` for full details.

You will use your `hash_list` implementation from the previous lab (`hash_list.h` and `hash_list.cpp`), and create a new `hash_map.cpp` file that implements the functions from `hash_map.h`. You are also responsible for creating your own `main.cpp` to test your implementation, although this file does not need to be submitted.

## Submission instructions

Upload your `hash_map.h`, `hash_map.cpp`, `hash_list.h`, and `hash_list.cpp` files to Gradescope. The version of `hash_list.h/cpp` that you upload MUST implement the iterator.

## Restrictions

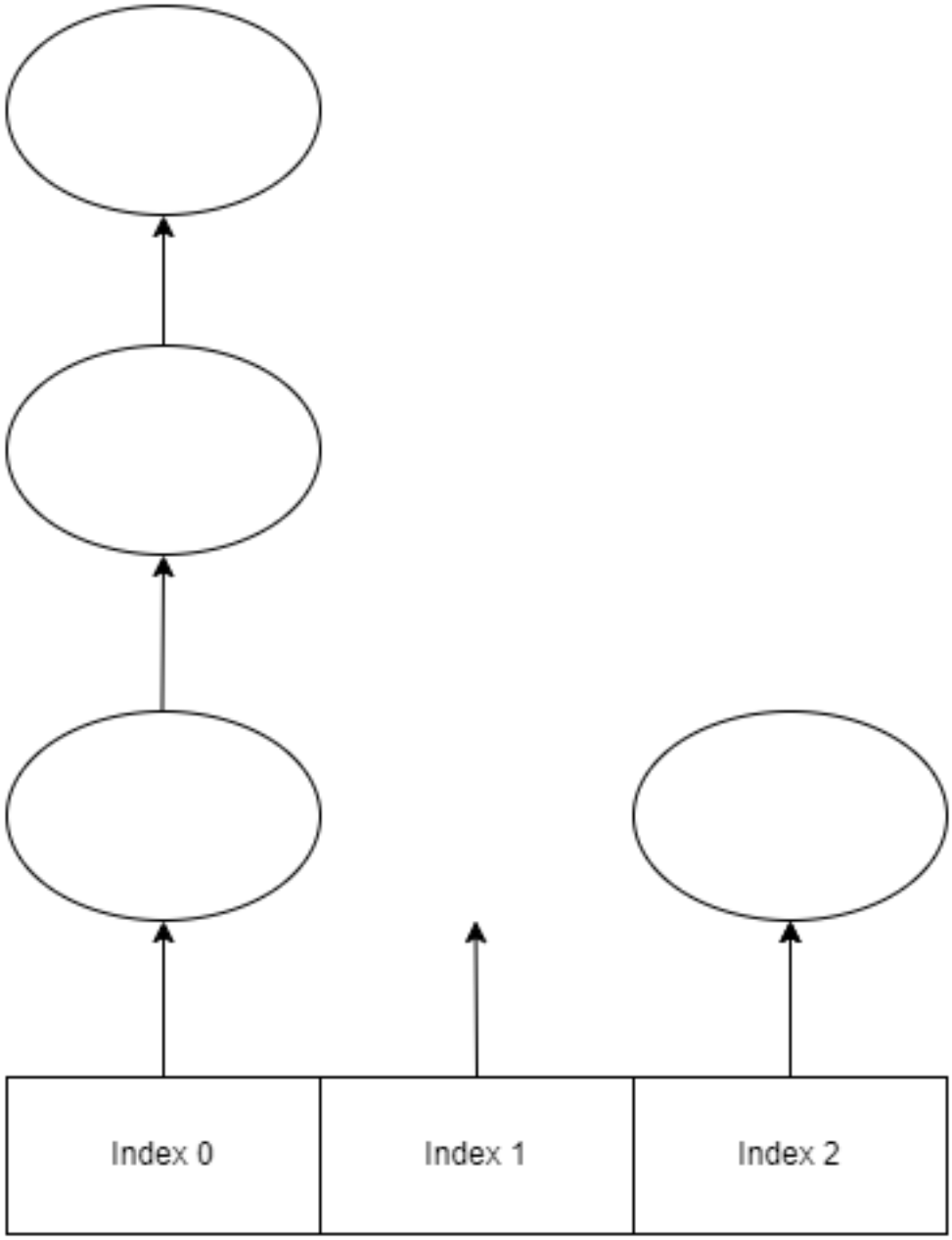
- Must submit all listed files
- Your `hash_map.cpp` file is only allowed to include hash_map.h
- Your `hash_list.cpp` function is only allowed to include hash_list.h
- You may not use any standard containers.
- You aren't allowed to delete any provided public/private functions/member variables from `hash_map.h`. You may add private or public functions/member variables if you want, but you can't remove anything from `hash_map.h` that was there when we distributed it to you.

## Hashing function

In theory you could use any hash function you wanted, but we're requiring you to use the absolute value of the key modulo `_capacity` (`_capacity` is defined in hash_map.h and is an argument to the constructor) as your hash function. If you don't use this hashing function then your `get_bucket_sizes()` implementation won't return the expected results and you won't get points for it.

## get_bucket_sizes explanation

Let's imagine we have a map that has a capacity of 3 (this means the array of hash lists is 3 items long). We want to know how many elements are in each of those hash lists




The get_bucket_sizes function needs to populate the index of the passed in array with the size of the list at that index. So using the above example we get

```
list at index 0 has size 3
list at index 1 has size 0
list at index 2 has size 1
```

So we would set the passed array to

```
-------------------------
|   3   |   0   |   1   |
-------------------------
```