Lab 7: Hashing
UWYO COSC 2030

Introduction:

As you might have guessed by now, major events in this class happen in three's. The third method for spellchecking is going to involve loading the dictionary into a hash map. Most of the specifics of the implementation of that will be taken care of by you in program three, but there's still time before that's due so for now we'll stick to a basic, critical component: the hash function itself. The general idea here, as previously discussed in the lectures, is to uniquely map a given word to a given integer, then store the word in a data structure at that value. This is where the concept of a key-value pair comes into play (the key being an integer and the value being the string). My goal in this lab is to get you to think about how you can limit collisions in your hash-map, as the more collisions you have, the slower your program will be.

Lab 7:

First, go ahead and accept the GitHub assignment here: https://classroom.github.com/a/Z7oxCjiY

There are two hash functions at the top of the .cpp file. Don't touch the one labeled "example", but edit "hasher" in any way that you would like. As you go, note the things that make the collisions decrease and the ones that make it increase. There's no real metric for what this should look like, except that your collisions should be better than the default. But you can go as far as you would like with this. For example, these are my personal program three results from when I took the class:

```
Time taken: 0.247597
Time taken (milliseconds): 247
There are 950068 words found in the dictionary
There were 1039082 compares. Average: 1

There are 27075 words not found in the dictionary
There were 5672 compares. Average: 0

There were 2544 words that weren't checked.
```

Submission:

Push your singular Lab7.cpp file and your README.md to GitHub, ***and nothing else***. This week I will take points for submitting extra files (yes, this includes a.out). Make sure you submit by April 13th at 11:59 pm.