In this assignment, I learned how to hash and work with buckets. Hashing is important because it allows you to get items in an array as fast as putting an item into an array. This is good because if you have an array of over 1 million items and you need to search through it, then it will take a long time. However, if you have some sort of algorithm that can help you search for the item, somehow to predict where it could be, then it would save a lot of time. There’s more time initially required to calculate where the item could be with the algorithm, but once you finish that, it’s very fast to just select that one individual item with an array.

One Issue I see with this, however, is that if you have buckets and multiple items are placed in one bucket, then you need to search if it’s there in the bucket. However, this may be circumvented with another hash. Of course, this would result in more space being used and this is not ideal when you have a million items. And then you get the issue where it’s possible that you may get buckets inside of buckets, which is still not ideal. An issue I had with the project was setting a starting table size in the CourseDBManager. You’re never told the number of items you need to insert into the CourseDBStructure, so you end up having to choose a large number. In my case, I chose 500, which is not ideal if you’re trying to conserve memory. I’m not sure there is a way to prevent this however without either recreating the array every time you add an item.

Buckets are good to use to make it faster to get items in an array, however they have their own drawbacks. The tests provided were not very good. When I created my tests, it took me some time to figure out how I needed to sort things, but I got it working in the end.