# CS 300 reflection

Amauri Hopewell

In this course, I focused primarily on use of data to solve the problem of reading information from a file, storing it, sorting it, searching through it, displaying results, and deleting items. This is an important problem to solve because it is the backbone of modern computer systems dealing with databases.

To approach this problem, I made use of data structures in two different ways. The first was the creation of custom classes or structs to store the unique data dealt with here, namely, course information. The second way was creating a data structure on top of that custom class or struct to allow leveraging pre-existing, well-explored algorithms for efficient manipulation, sorting, and searching. Understanding both types of data structures is important to enable efficient work on large databases.

Some of the roadblocks that I had to overcome when dealing with this were the wide variety and choice of data structures available, and their sometimes unusual implementation in C++. To overcome this, I tried to make use of standard, well-documented libraries as much as possible. I also opted for simplicity when possible given that many of these data structures have been extraordinarily well optimized in the standard libraries. For instance, though a vector might on its face appear to be less efficient than a hash table, C++ has so many behind-the-scenes optimization that it often performs like a hash table.

My work on this project in this class has changed the way I design and develop software by opening my eyes to the wide range of data structures that are out there, with efficient and standardized algorithms. It has also made me pay attention to what the underlying data structures in a library might be so I can better understand which ones are likely to be efficient.

It has also helped me write more modular, readable, adaptable, and maintainable code. This is because leveraging these common data structures is like speaking a new language that most programmers know, and so framing a solution in terms of these data structures can make them much easier to maintain, while avoiding re-inventing the wheel.