//==============================================

// Name: Jungjoo Kim

// Student Number: 162 641 195

// Email: jkim594@myseneca.ca

// Section: NAA

// Date: July 15, 2021

// Workshop: Workshop 7

//==============================================

I had an opportunity to use various STL functions through this workshop. First, in this workshop, we used the **vector** to store Song structures in the main variable m\_song like “std::vector<Song> m\_song;”. I had to read the file line by line and save the stored data to the Song structure and add it to the vector. In this process, vectors helped store new Song structures while variably increasing the size, even though I did not initially set the size. Another sequential container, **list**, was also used because to store the corresponding Song structure I found like "std::list<Song> result;". Obviously, there is a difference between a vector and a list. The way to access elements in containers is different and functions are different. However, in this workshop, I did not significantly use the characteristic of the list such as add and delete elements quickly. If I use the vector or list container, I must include <vector> and <list>. Lastly, I use several functions in <algorithm>. It used a **sort** algorithm. The reason was to align a vector container in the order the user wanted. The iterator representing the range of elements was used, and the expression of how to align them used the Lambda formula. The **for\_each** feature was also used to run something in the same way to each element while circulates the vector sequentially. For example, in function "cleanAlbum", found elements whose album title is None value and made the album title empty. The reason I used **any\_of** was to find out if the vector container had the same value as the value of the parameter received. Finally, I used **copy\_if**.

"copy\_if(m\_song.begin(), m\_song.end(), back\_inserter(result), [src](const Song& theSong)". The reason was to find only the songs of the artist that the user wanted in the vector container and save them in the new container, List. But, I used the iterator, another STL function here. I included the <iterator> to use **back\_inserter**. The reason is that copy\_if does not automatically increase the size of the container to copy elements, such as push\_back. Therefore, to save Song structures to a new list via copy\_if, the list must be sized first. However, back\_insertor automatically increases the size like push\_back, allowing elements to be inserted into the end of the container.

There are several advantages of using the STL algorithm rather than the basic loop function. First, code can be implemented inline. Inline code is legible and neat to look at. This means that it is extremely beneficial for code review and debugging. And if I did not know the STL function, I would have done it by implementing all the functions line by line, such as sort and copy\_if, but the STL function solves it at once. That is, it adds tremendous productivity to code creation.

In the process of receiving a file name and instantiating the SongCollection class, an error will occur if the file name is incorrect. At this time, it is very important to detect errors and deliver messages to users. The method that can be used is try-catch. Try reading the file using the file name the user entered, and if the file does not open, use throw to send an error message.