**CSCE 4523 Database Management Systems**

**Homework 1**

**By: Max Thursby and Kevin Zheng**

**Objectives**

The objective for this assignment was to implement a database system using file management techniques. We will implement a simple database using a sorted file of fixed length records. The files we will use to accomplish this goal will be the Titanic.csv and SmallTitanic.csv. We will have to complete the methods of the database class to manage the database.

**Approach**

The way we implemented this was to go over the sample code and see what was provided to us and see what methods we can use to complete the database management system. We then added a menu system for the user so they can have control on what they want the program to be doing. We then added a switch statement for the menu choices and then added in the specific methods to allow the correct functions. We would add in things like created, open, and close database methods to the correct cases. For open database we would check to see if there is already a database opened. For read, display, update, delete, and add we would add outputs and inputs to ask for the users inputs and the users can then input a possible choice. The language we used for this assignment was C++. Strategies were to not make too many methods than necessary as it could get confusing on what is what. Some Design choices were to make things readable and easy to implement as we would know where things are ordered. The format we choose for the record is id, first name, last name, age, ticket number, fare, and purchase date in that order. Also when reading in the csv file we will first put the record then an empty record until all the data from the csv is added to the data file. The size for each field is 7 for the id, 12 for the first name, 20 for the last name, 3 for the age, 20 for the ticket number, 6 for the fare, and 10 for the purchase date. The delimiter for our program are commas. The total record size is 85 for linux and 86 for windows. The config file displays the numbers of records and record size. Config file will be part of the program folder. The sample code we started off with were the given sample codes and solutions for C++ from the homework 1 instructions. We prohibit the user from adding overflow files to the database.

**Results**

Some error handling needed for this assignment was to check if the numbers the user entered for the menu was in range. If it isn’t in range it would continue to keep asking for an input until it’s one of the options. This was also the same case for the update menu too when asking on what field to update. Also for the create report we would also check to see if it’s less than ten reports so it won’t go into an infinite loop. More error handling is making sure the updated field is being updated correctly on the picked record and also when deleting the record making sure it’s deleted in the right place and correctly. The efficiency is making sure the program ran well with no bugs and also making sure it was readable in order to fix bugs when there were any. The things that worked well is making sure our methods were doing what they were intended to do and also making the code readable. This helped for further development and didn’t make any big confusions. In this assignment Max did the creating database, opening, update, and the add record. functions for the report. Also Max edited all the sizes for each field. Kevin did some of the bug fixes when there were any and also did the closing database, display, delete, and create report.

**Testing**

The way we tested the program was to test each method we implemented one at a time so it was easier to fix bugs if there were any. We would also enter in varied inputs to check if there were any bugs in the program. Referring to the typescript everything worked as intended. From the test cases it displayed expected output shown from the test case instructions. Input values are in a separate file from the typescript

**Typescript**

Typescript will be uploaded separately.