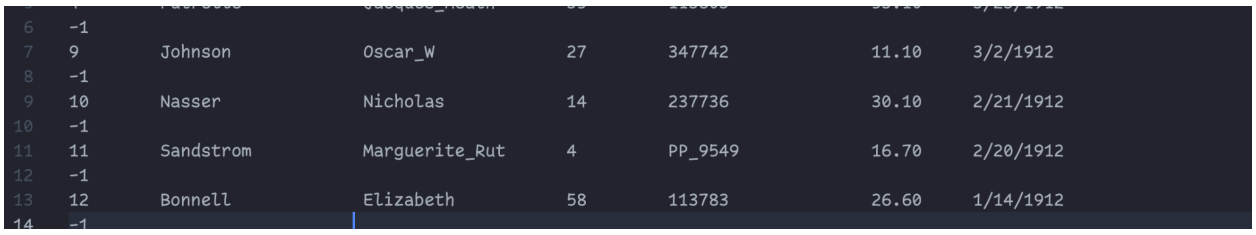
CSCE 4523 Database Management Systems
Homework 1
By: Phuong Hoang Tran (Alex)

Objective

The aim of this assignment is to learn how to handle files and create a simple database system. We will work with data from the Titanic, organizing it into a database using code.

Approach

To implement this simple database system I have chosen C++ due to its efficiency for working with large text files. To make my database adaptable for other database table schema, I leverage the factory design pattern and dependency injection to prevent tight coupling between my record class and database class. Following the dependency inversion principle also helps with code maintainability in the future, all I have to do to swap out a record type is change one line of code. As for saving the record to a text file, the order of the field doesn't matter because I replace spaces with underscore in each field and use spaces as my delimiter. My total record size is 100 bytes per record including the new line character. At first I used the sample code, but for homework 2 I decided to rewrite all the methods and redesign the entire thing.



```
6   -1
7   9       Johnson         Oscar_W         27      347742          11.10   3/2/1912
8   -1
9   10      Nasser          Nicholas        14      237736          30.10   2/21/1912
10  -1
11  11      Sandstrom       Marguerite_Rut  4       PP_9549         16.70   2/20/1912
12  -1
13  12      Bonnell         Elizabeth       58      113783          26.60   1/14/1912
14  -1
```

Result

For error handling, the basic fundamental of ensuring that user input is valid is not implemented, instead I focus more on the database method edge cases. The fundamental logic of ensuring that the database is opened before any CRUD operation is implemented. Along with that are some out of bound checking.

Testing

At first, I decided to use unsigned int for everything, this was a bad idea because it required a significant amount of error checking so I switched back to using int. Unsigned int operates normally until the number gets smaller than 0, instead of the number being negative, it loops back and becomes the maximum which causes my binary search function to freak out and explode.