## CSCE 2004 - Homework 7 Due Date – 05/04/2023 at 11:59pm

## This project is to be done on Turing.

## **Problem Statement:**

The objective of this programming assignment is to bring together several key programming concepts (file input/output, arrays, and classes) to build an interactive program for looking up information about famous computer scientists. To initialize the application, students must read an ascii file called people.txt (included in blackboard). Each line of the file contains the following pieces of information:

* + first\_name
  + last\_name
  + birth\_year
  + death\_year
  + contribution
* If you look at the people.txt file, you will notice that the first and last names are single strings, and that the birth and death years are integers. A death year of -1 is used to indicate that the person is still alive. The contribution field is a string with a variable number of words. Therefore, you will have to use getline to read to the end of a line in order to read in the contribution.
* Your task is to read/store the information above into an object oriented data structure. Your program should then allow the user to search this data structure for computer scientists three ways: (1) using their first name, (2) using their last name, or (3) using a range of birth years. Your program should print out all records that match the search criteria.

### **Design:**

Students are required to define two classes "Person" and "Table" to contain the information in the "people.txt" file. Each line of the input file corresponds to a Person to be inserted into into the Table. Therefore, the data contained in each line should be used to set a Person object's data members. The Table class should contain an **array** of Person objects. Students must define these two classes and specify the constructor destructor functions, the public methods and private variables.

* The Person class must contain the following methods.
  + constructor - to initialize the object.
  + copy constructor - to copy an object.
  + destructor - to delete the object.
  + set - allow the user to set the properties of a person.
  + get - allow the user to get the properties of a person.
  + print - print out the record.
* The Table class must contain the following methods.
  + constructor - to initialize the object.
  + copy constructor - to copy an object.
  + destructor - to delete the object.
  + print - print out all of the records as a nicely a formatted table.
  + read - read a list of persons from a file formatted as above.
  + search by name - allows the user to search the table for a name. When a name is found, the person's print method should be called.
  + search by year range - allows the user to search the table by a birth year range (upper and lower bounds). If a person's birth year falls into a given range, then the person's print method should be called.
* The methods above should be used to implement the interactive search program. Students may want to refer to previous labs for examples of file input/output and classes.

### **Implementation:**

You can implement this program using either a bottom-up approach or a top-down approach. If you go for a bottom-up approach, start by creating basic methods and classes, and test these methods using a simple main program that calls each method. When this is working, you can create the main program that uses these methods to solve the problem above.

If you go for a top-down approach, start by creating your main program that reads user input, and calls empty methods to pretend to solve the problem. Then add in the code for these methods one at a time. This way, you will get an idea of how the whole program will work before you dive into the details of implementing each method and class.

Regardless of which technique you choose to use, you should develop your code **incrementally** adding code, compiling, debugging, a little bit at a time. This way, you always have a program that "does something" even if it is not complete.

### **Testing:**

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with a range of search commands (including several where data is found in the table, and several where no information is found) and save your testing output in text files for submission on the program due date.

**To compile:**

g++ -Wall hw7.cpp person.cpp table.cpp -o hw7.exe

**To run:**

./hw7.exe

**Create a typescript (a log file)**

The following will generate a typescript of your program outputs:

username@turing:~$ script

username@turing:~$ g++ -Wall hw7.cpp person.cpp table.cpp –o hw7.exe

username@turing:~$ ./hw7.exe

username@turing:~$ exit

Be sure to include **at least 2 test cases** in your typescript.

### **Documentation:**

When you have completed your C++ program, write a short report (less than one page long) describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report in a separate text file to be submitted electronically.

### **Project Submission:**

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic analysis of all programs that are submitted. When you have completed the tasks above go to the class web site to "upload" your documentation and testing document (pdf, doc, or txt file), and your C++ program (cpp or txt file).

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

* + 10% off if less than 1 day late,
  + 20% off if less than 2 days late,
  + 30% off if less than 3 days late,
  + no credit if more than 3 days late.
* You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.