CS&E 1222

Lab 13 – Classes & Write to Files Lab Assignment – 20 points

- ✓ The *lab* must be accomplished solely by you:
  - ➢ DO NOT look at anyone's code other than your own, including code from another's student in your section or another section of the course, or any third party source, e.g. the Internet
  - ➢ DO NOT share or copy anyone else's code for any graded assignment
  - ➢ DO NOT work in pairs or groups
- ✓ All cases of academic misconduct will be reported to the *Committee On Academic Misconduct* (COAM).

Setting up the Programming Environment

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the lecture notes. Also see the example code from Lab #1. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments. ***In addition, read the document on "Commenting" found in the Content tab on Carmen under "Tutorials"***.

1. At the Linux command line type `mkdir lab13`. This will create a new directory named **`lab13`**. Work out of this directory. In order to do that, type `cd lab13`. This changes the current working directory to the directory **`lab13`**.
2. If you have created the directory **`lab13`**, then just type `cd lab13`.

3. Copy the file **`temps_template.cpp`** by typing

```
cp /class/cse1222/9643/lab13/temps_template.cpp temps.cpp
```

4. Copy the file **`temps_solution.exe`** into the current directory.

```
cp /class/cse1222/9643/lab13/temps_solution.exe .
```

Be sure to include ***9643*** (this is your course section indicator) and the period, ".".

Programming Assignment

Write a program that will collect from the user a list of temperature samples taken throughout a single day. Each temperature sample consists of a temperature (degrees Fahrenheit) and a time (military time in hours and minutes only). The hours in military time is an integer between 0 and 23 and the minutes is an integer between 0 and 59. Your program will display military time in the format HH:MM, where HH is the hour and MM is the minutes. For example, midnight is 00:00 in military time, noon is 12:00, 9:10am in standard time is 09:10 in military time, and 3:15pm in standard time is 15:15 in military time.

The procedure main() (see the code template **`temps_template.cpp`**) has been written for you (DO NOT change anything in the main() procedure) as well as the members (attributes and functions) of the Classes **Miltime** and **Fahrenheit**. You will write code for the member functions of both classes and also helper functions as indicated by the comments */* INSERT CODE HERE */*.

Run **`temps_solution.exe`** to see an example of the program.

A sample run of the program would look like this (bold indicates what the user will enter):

> **temps.exe**
Enter the output file name: **out.txt**
Enter the number of samples: **2**

Enter degrees(Fahrenheit): **212**
Enter length: **4**
Enter height: **27**

Enter degrees(Fahrenheit): **32.5**
Enter length: **0**
Enter height: **1**

> **cat out.txt**

Sample #1: 212 degrees F (or 100 degrees C) at 04:27
Sample #2: 32.5 degrees F (or 0.277778 degrees C) at 00:01

The average temp is 122.25 degrees F
The coldest temp is 32.5 degrees F
The last sample was taken at time 04:27

To view the contents of the file, type "**`cat out.txt`**". Your program must work for any file names besides "**out.txt**".

1. The function prototypes are provided for you and must NOT be changed in any way.
2. The procedure main() is provided for you and must NOT be changed in any way.
3. Understand the class definitions for **MilTime** and **Fahrenheit**.
4. Understand the algorithm in the procedure main().
5. Do **NOT** add more functions/procedures to the solution.
6. Each function should have a comment explaining what it does.
7. Each function parameter should have a comment explaining the parameter.
8. Write code by replacing every place */* INSERT CODE HERE */* appears with your solution.

9. Implement the member functions for the class **MilTime**. Use the appropriate class attributes of the class **MilTime** in your solution.
    - Use the provided class attributes in your solution of the class **MilTime**.
10. The member function write_out() of the class **MilTime** writes the military time in the format ***HH:MM*** to a given file output stream. Write code to ensure this format.
11. You may assume that the user will enter only valid values for hours (0-23) and minutes (0-59).
12. Implement the member functions for the class **Fahrenheit**. Use the appropriate class attributes of the class **Fahrenheit** in your solution.
    - Use the provided class attributes in your solution of the class **MilTime**.
    - Include the C++ command "#include <cstdlib>" to use the C++ exit command.
    - CLOSE the file stream. (Note: You will lose points if you forget to close the output file stream, even though your program runs correctly).
    - Important note, do not write redundant code.

13. The member function getTemp() of the class **Fahrenheit** retrieves the degrees stored in Fahrenheit.
14. The member function getTime() of the class **Fahrenheit** retrieves the military time (an object).
15. The member function getCelsius() of the class **Fahrenheit** converts the stored degrees in Fahrenheit to Celsius and returns this conversion.
16. The member function write_out() of the class **Fahrenheit** writes a list of the samples and other information to an output file given a file name.
    - Use the file name to open the file for output. Review and use the program writeFile3.cpp discussed in class as a template for your program. (See the class slides or download the program from Carmen). Make sure to pass a C style string, not the C++ string, to the open routine. Be sure to use the type "ofstream", not "ifstream", for your output file streams. Check if the file was successfully opened for output. If not, ***print an error message*** (see sample executable) and exit. The "**#include <cstdlib>**" allows you to C++ **exit** command.
    - CLOSE the file stream. (**Note: You will lose points if you forget to close the output file stream, even though your program runs correctly**).

17. These functions are called from the main() function.
    - Implement the function read_filename() that prompts, reads, and returns a file name as a string. Include the C++ command "#include <string>" to use C++ strings.
    - Implement the function read_num_samples() that prompts and reads the number of temperature samples (degree and time) that the user will enter.
    - Implement the function read_samples() that prompts the user for a single temperate sample and then returns a newly constructed object.
    - Implement the function write_to_file() that display all output to the output file. This function must call the following functions in order to receive full credit:
        - Implement the function average_temp() that computes the average temperature over all of the samples.
        - Implement the function coldest_temp() that returns the lowest temperature value over all of the samples.

- Implement the function last_sample() that returns the last temperature sample taken during the day.

18. TEST YOUR CODE THOROUGHLY. For example, pay close attention to the format of the output including empty lines. Your program must not have a run-time error.
19. Be sure to add the header comments "File", "Created by", "Creation Date" and "Synopsis" at the top of the file. Each synopsis should contain a brief description of what the program does.
20. Be sure that there is a comment documenting each variable.
21. Be sure that your if statements, for and while loops and blocks are properly indented.
22. Check your output against the output from the solution executables provided.

Submit Your Work

Important: Any program which does not compile and run will receive no credit!
If you are not sure what this means please ask your instructor.

Submit the file **`temps.cpp`** using the *Lab 13* drop box on Carmen. **DO NOT** submit the file **`a.out`**. **DO NOT** submit uncompleted work from *Quiz 13*. This will not be graded.