Final Assessment – Part 2

*(Worth: **7.5%** of your final grade)*

Introduction

This is part-2 of the final assessment and has two sections:

1. **Source code** solution to a problem described further on in this document
   - **Worth 40% (graded out of 3.0)**
   - This section may be done individually or in groups (no more than 3 students per group).
   - If you plan to do this section as a group, you must receive authorization from your instructor before starting.
   - If this section is done as a group, all members will receive the same grade (for this section)

2. **Reflection** containing answers to specific topics/concepts as directed in the reflection section described further on in this document
   - **Worth 60% (graded out of 4.5)**
   - This part must be done by all students individually

Submission Policy

All work (including reflections) must be submitted to your instructor no later than**:**

Friday August 14th before NOON (11:59 AM EST)

Your Digital Signature

- **Every file submitted (.c | .h | .txt | .docx) must contain at a minimum, your full name and Seneca email address**
- **If you are submitting as a group, each member's information should be included**

By stating your identity on each piece of work, you are authenticating you/your group are the author(s) of the material and that you have not plagiarized (copied) anyone else's work

Note: You are responsible for backing-up your work regularly.

Your Task – Source Code (40%)

Download or clone the final assessment part-2 (**Final-P2**) from **https://github.com/Seneca-144100/Final-Part-2**

The X Country Ski Race

You have been tasked with developing a program to read the information about a X country ski race from a text file and produce a report on it. The race is a major event and could attract up to 2,000 participants. The potential of having such high volume of data will require you to be efficient in how the data is read, stored and processed. There are three timestamps recorded for each skier:

1. The start time
2. The time when the skier reached the midway point
3. The completion time (or the time at which the skier withdrew)

In the event that a skier does not complete the course, they are marked as having not completed and the time at which they withdrew is recorded but not used in any calculations and skiers who do not finish are not eligible for any awards.

Skiers are organized into categories based on **age** and the **race length**. Age groupings are determined by age ranges, where **16-20** are **juniors**, **21-34** are **adults** and **35 or older** are **seniors**. There are three different races lengths, **S** (**short, 10km**), **M** (**medium, 25 km**), and **L** (**long, 50 km**). The race will have different start times, but this has no impact on the durations it will take the skiers to complete the course.

You cannot guarantee that enough skiers will enter for each distance category to win all awards available and awards that are not awarded will be listed as "**Not Awarded**".

For example:

1. If you request a report for the winners of each distance category and there is a category with no entrants, you would list it as "Not Awarded" rather than putting a skier's name.
2. Likewise, if you request a report for the top 3 skiers in each distance category and a category only has 2 entrants, then you just print the top 2 entrants.

The following is a sample of the file "**data.txt**" which will be read by the program. Each line of the file contains the name of the skier, age, race length (S, M, or L), the clock time of the race start, clock time of the midway mark and clock time of finishing the race. In the event that a skier withdraws, there is a "W" at the end of the line and the midway time will be zero if they did not make it that far while the finish time will be when the skier withdrew.

====================== Sample "data.txt" File ======================

```
Eddy Mercx, 72 M 1:10 1:59 2:58
Jocelyn Lovell, 60 M 1:10 0:00 1:34 W
Jason Gaudet, 28 M 1:10 2:11 3:09
Claude Van Gogh, 20 M 1:10 2:25 3:24
Lance Armstrong, 40 M 1:10 2:02 3:05
Arlenis Sierra, 26 L 1:00 1:51 2:41
Billy F. Gibbons, 62 S 1:20 2:42 4:28
Charlie Watt, 66 S 1:20 3:08 5:39
Cecilie Ledwig, 23 L 1:00 1:49 2:38
Nikki Sixx, 48 S 1:20 2:59 4:58
Kirsten Wild, 34 L 1:00 1:55 2:54
Eddie Van Halen, 63 S 1:20 2:10 3:55
Rachel McKinnon, 37 L 1:00 1:39 2:41
Angus Young, 61 S 1:20 2:02 3:10
```

====================== End Sample Data File ======================

***NOTE**: Remember this data file could likely contain a maximum 2,000 records*

- The above sample "**data.txt**" file is included and should be used in your development.
- You should take a backup of this file so you can modify it with different data to thoroughly test your reporting logic (not all scenario's are demonstrated in the example)

File Helper Module

A "**file_helper**" module has been provided for you and your code must use the provided ***readFileRecord()*** function in order to read each line of the data file.

The code requires the use of a ***SkierInfo*** structure. The ***SkierInfo*** structure members are purposely missing and will require you to define them based on the code in the ***readFileRecord()*** function.

NOTE: You are not to modify the "file_helper" module files with the exception of completing the SkierInfo structure members in the "file_helper.h" file.

Additional Module's

You are free to design your solution as you see fit, however, your solution design should implement additional modularity if/when possible.

Application Main Entry Point (main)

You will be required to develop the main function (application entry point) in the file "**main.c**" that will launch your solution.

Menu & Reporting Options

The following menu/reporting options must be offered:

- **"Display the best 3 skiers in a distance category"**
  Note: "best" means the fastest completed times (order lowest to highest duration)
- **"Display all skiers in a distance category"**
- **"Display the worst 3 skiers in a distance category"**
  Note: "worst" means the slowest completed times (order highest to lowest duration)
- **"Display the winners in all distance categories"**
- **"Exit the application"**

The following two pages, is an example output execution of the minimum expected behaviour of how your solution should work.

Note

- You are encouraged to improve on the menu in a way you think would be more user-friendly
- Your solution should closely (if not exactly) match the tabular format used in the example reporting output/display parts (must be aligned)
- **The example does not show all possible data output scenario's described in the requirements such as the scenario for "Not Awarded". However, it is expected your solution will work and behave as required in all possible cases**.

Example Execution

Highlighted text represents user input.

```
********** X-Country Ski Race Reporting **********
What would you like to do?
0 – Exit
1 – Print top 3 skiers in a category
2 – Print all skiers in a category
3 – Print last 3 skiers in a category
4 – Print winners in all categories
: 1

Which category (S, M, L): s

Skier                    Age Group Time
---------------------------------------
Angus Young                 Senior 1:50
Eddie Van Halen             Senior 2:35
Billy F. Gibbons            Senior 3:08

What would you like to do?
0 – Exit
1 – Print top 3 skiers in a category
2 – Print all skiers in a category
3 – Print last 3 skiers in a category
4 – Print winners in all categories
: 2

Which category (S, M, L): S

Skier                    Age Group Time Withdrew
-----------------------------------------------
Angus Young                 Senior 1:50       No
Eddie Van Halen             Senior 2:35       No
Billy F. Gibbons            Senior 3:08       No
Nikki Sixx                  Senior 3:38       No
Charlie Watt                Senior 4:19       No
Jocelyn Lovell              Senior  N/A      Yes

What would you like to do?
0 – Exit
1 – Print top 3 skiers in a category
2 – Print all skiers in a category
```

```
3 – Print last 3 skiers in a category
4 – Print winners in all categories
: 3

Which category (S, M, L): S

Skier                    Age Group Time
-------------------------------------
Charlie Watt                Senior 4:19
Nikki Sixx                  Senior 3:38
Billy F. Gibbons            Senior 3:08

What would you like to do?
0 – Exit
1 – Print top 3 skiers in a category
2 – Print all skiers in a category
3 – Print last 3 skiers in a category
4 – Print winners in all categories
: 4

Skier                    Age Group Category Time
-----------------------------------------------
Angus Young                 Senior    10 km 1:50
Eddy Mercx                  Senior    25 km 1:48
Cecilie Ledwig               Adult    50 km 1:38

What would you like to do?
0 – Exit
1 – Print top 3 skiers in a category
2 – Print all skiers in a category
3 – Print last 3 skiers in a category
4 – Print winners in all categories
: 0

Keep on Skiing!
```

Reflection (60%)

Each student must submit a reflection and will be independently graded (if you are working in a group, only the source code portion will be a shared grade). Please provide answers to the following in a **Microsoft Word** or **raw text file** named using your **last name followed by your first name**. Remember, include your **DIGITAL SIGNATURE** as described on page one of this document.

Your reflection answers should be detailed and accurate. A suitable response should be **at least 125 words** for **each** reflection question. This implies your **overall** reflection must be a **minimum of 375 words (not including the question if you repeat it in the answer)**.

- This program can greatly benefit from modular programming. Describe how you decided to make your program more modular. For each function **you created**, describe your decision to make the function and why its contents should be in one function.
- The full-text for the distance categories and age groups is repeated a lot. Describe how you managed this – did you store this efficiently to reduce wasting memory? Why is the technique you used more space efficient than other techniques? Pick one other technique that is less efficient than your method and describe why your method is superior (better).
- Reading the data from the file presented some challenges. Explain how the readFileRecord() function determined when the end of line (record) had been reached, given that some of the skiers might have withdrawn and would have a different number of fields on the line. Describe another technique you could use to handle the differing number of fields on each line (record).

Submitting Your Work

Source Code

Create a **Microsoft Team**:

- Using the following naming convention:
  "2020-Summer-IPC144(**NAA**).**lastname.lastname.lastname**"
  (*replace "**NAA**" with your section, "**lastname**": each member's last name*)
- **Add your instructor** as a "**Member**"
- Post your source code in the "**Files**" section of the team
- **Remove or move** to a sub-directory, any files that **should not be included** with your submission
- Your instructor will download your source code upon the deadline

Groups
- Designate ONE student to post the files for the group
- Be sure all files have a DIGITAL SIGNATURE that includes **all members**

Reflection

Post your reflections to Blackboard. Look for the assessment called "***FinalP2Reflect***". The window of availability will be from **August 5th (8:00 AM EST)** through to **August 14th at NOON (11:59 AM EST)**.

Note
- The reflection file should be **named with your full name (last name first)**
- Your reflection can be stored as a **Microsoft Word**, or a raw **text file**.

Grading Rubric

The source code will be graded using the following rubric:

Source Code (40% of total mark)

Correct Solution to problem	30%
Appropriate data structures	20%
Modular programming used	20%
Efficiency of coding	20%
Quality of code	10%

The reflection will be graded using the following rubric"

Reflection (60% of total mark)

Correctness	60%
Sufficient (enough) detail	30%
Grammar	10%