CS 341 – Spring 2023
Assignment #1 – Keepin' It Classy
Due: 1/23/2023

This first assignment will serve as a refresher for the C++ programming language and allow you to become more comfortable with the tools we will be using this semester to accomplish our goals. This project will serve as the foundation for exploring the many benefits that the C++ programming language can provide and will help you re-familiarize yourself with Object-Oriented concepts that you previously learned in CS 248 (in Java) all the while learning `Make` and `Git/GitHub` in the process.

For this assignment, since we live in the racing capital of the world, you will be simulating the world famous Indianapolis 500. *Ladies and Gentlemen start your engines…*

Development Process:

Phase I

Your program will take the field of cars (provided in a text file on Canvas named: `Files->Assignments->A1->indy500.txt`) and begin by reading in the contents of the file. You will use standard File I/O in C++ to accomplish this task (discussed in the lecture slides). The structure of the text file will be as follows `<Name> <Speed> <Control>`:

```
Kanaan 6 9
Castroneves 6 10
```

The file will contain thirty-three (33) lines of data for you to read-in. Your job will be to create a `driver.cpp` (no pun intended here…) and a `main` function that accomplishes this task. Once you have completed this you can test your program by printing out in the console the contents of the text file. Success will mean you have completed Phase I and are ready for Phase II...

Phase II

When you are completing Phase I you are likely asking yourself…"*what are we going to do with all this data?*" That is where Phase II comes in – you will take the data collected in Phase I and build a unique instance of a `Car` for each line read-in. Here you will explore OO programming in C++. As part of this phase you will create two files: `Car.cpp` (definition(s)) and `Car.h` (declaration(s)). We will explore, in lecture, how to accomplish this task.

Your `Car` class will contain the following three (3) attributes: `name`, `speed`, `control`. The `name` attribute will be of type string and hold the name associated with the individual racecar `<Name>`. The `speed` attribute will be of type integer and hold the speed value associated with the individual racecar `<Speed>`. The `control` attribute will be of type integer and hold the control value associated with the individual racecar `<Control>`. In addition to these three (3) attributes your class **MUST** also contain appropriate accessor methods along with a default constructor and destructor. You may either use an additional method to construct a `Car` object or create a second constructor to accomplish this.

Once you have constructed all thirty-three (33) racecars in the field you can test your program by creating a `printInfo()` method that prints out each `Car` objects information. In order to successfully compile your Car class and your driver.cpp you will need to create a `Makefile` (discussed in the lecture slides). This `Makefile` should generate an executable file named `Race.exe`. You will then run `Race.exe` at the command prompt. Success will mean you have completed Phase II and are ready for Phase III – **THE GREATEST SPECTACLE IN RACING!**

Phase III

Here we are going to simulate the Indianapolis 500 mile race. Before we begin the race we need to make some last minute tweaks to our existing program – think of it as Carb Day.

We will begin this phase by adding a new attribute to our `Car` class (`lapNumber`) – don't forget to add the appropriate accessor methods to go along with it! This will allow each `Car` object to keep track of what lap they are within the race. All racecars will begin with a `lapNumber` value of 0. We will also need to create a new function on our driver.cpp that will start the race (`simulateRace()`). This function will be responsible for simulating the race. In order to simulate the race we will use the two attributes in conjunction with a random number generator (as discussed in lecture) to "advance" the `Car` objects each lap. If the racecar has a `control` value higher than '5,' we will take that number and divide it by a random number between 1 and 10. We will then add that number to their `speed` value and that will be how many laps they advance each "turn." If their `control` value is 5 or lower we will simply increase their `lapNumber` by their `speed` value. The race will consist of 200 laps (500 miles). The first `Car` object to reach lap 200 will be declared the winner and the race will terminate. Your program should display the winner of the race along with each racecar and their current lap.

A sample of final output is shown below:

```
***Welcome to the 2023 Indianapolis 500 Race!***

***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...
***--> Racing...

        ***Carpenter has won the 2023 Indianapolis 500! ***
```

```
Driver Name: Lap Number
**********
Kanaan: 182
Newgarden: 154
McLaughlin: 154
Kellett: 110
O'Ward: 110
Montoya: 175
Castroneves: 195
Rosenqvist: 110
Ericsson: 110
Dixon: 199
Palou: 105
Hildebrand: 160
Power: 195
Kirkwood: 84
Rahal: 105
Malukas: 63
Daly: 154
VeeKay: 120
Ferrucci: 105
Karam: 105
Wilson: 105
Herta: 126
Rossi: 170
Grosjean: 84
DeFrancesco: 63
Lundgaard: 63
Carpenter: 200
Harvey: 105
Johnson: 113
Sato: 177
Pagenaud: 176
Ilott: 63
Andretti: 156

Thank you for using my race simulation program - goodbye!
```

Submission:

All assignments must be submitted on Butler GitHub (github.butler.edu) – this will be discussed in the lecture slides if you are unfamiliar with this process. This an **individual** assignment – meaning that each student should submit their OWN work. The name of your Butler GitHub repository must be as follows: **cs341_spring2023**

For this assignment, all development must take place on the **master branch** in your GitHub repository. It is strongly recommended that you commit and push often! Please make sure to commit for each Software Development Phase – this will allow me to check on your progress and provide helpful tips! By applying this process in this assignment and practicing it will help to familiarize you with the workings of a source code repository and its importance in software design and development. We will be using this the entire semester – so best to become very well versed in it now!

The directory structure of the repository must contain the following files:

- **driver.cpp**
- **Car.cpp**
- **Car.h**
- **makefile**

Each source file (.cpp/.h) **must** include the Honor Pledge and digital signature – more details about this can be found in the lecture slides. Additionally, you should use good programming practices throughout your program including comments, variables naming, indenting, etc. A portion of your grade will depend on this!