CSCI 1300 – Introduction to Computer Programming
Instructor: Hoenigman/Lewis
Assignment 7
Due Friday, November 6 by 8am

For this assignment, include all of your code in one file, called main.cpp, which is the default source file name in a CodeBlocks project.

**Submitting Your Code to Moodle**
You must submit your code to Moodle to get full credit for the assignment, even if the computer science autograder gives you a perfect score.

Please also include comments in your code to describe what your code is doing. Comments should also include your name, recitation TA, and the assignment and problem number. TAs will be checking that you code has comments.

**Submitting Your Code to the Autograder:**
The computer science autograder, known as COG, can be found here:
https://web-cog-csci1300.cs.colorado.edu

Login to COG using your identikey and password. Select the CSCI1300 - Assignment #07 from the dropdown. Upload your .cpp file and click Submit. **Your file needs to be named main.cpp for the grading script to run.** COG will run its tests and display the results in the window below the Submit button. If your code doesn't run correctly on COG, read the error messages carefully, correct the mistakes in your code, and upload a new file. You can modify your code and resubmit as many times as you need to, up until the assignment due date.

Before you submit your code to COG, make sure it runs on your computer. If it doesn't run on the VM, it won't run on COG.

If you do not get your assignment to run on COG before the assignment deadline, you will have the option of scheduling an interview grade with your TA to get a grade for the assignment. We'll talk more about scheduling the interview in lecture and recitation. Even if you do get the assignment to run on COG, you can schedule the interview if you just want to talk about the assignment and get feedback on your implementation.

**What to do if you have questions**
There are several ways to get help on assignments in 1300, and depending on your question, some sources are better than others. There is a Peer Discussion Forum on our Moodle page that is a good place to post technical questions, such as how to get user input, or treat that input as an integer. When you answer other students' questions on the forum, please do not post entire assignment solutions. The CAs are also a good source of technical information. If, after reading the assignment write-up, you need clarification on what you're being asked to do in the assignment, the

TAs and the course instructors are better sources of information than the discussion forum or the CAs.

## Football: Calculate the Passer Rating

1. In football, there is a statistic for quarterbacks called the *passer rating*. There are five input parameters to the calculation: pass completions, pass attempts, total passing yards, touchdowns, and interceptions.

   Write a program with five user inputs:

   i. Pass completions
   ii. Pass attempts
   iii. Total passing yards
   iv. Touchdowns
   v. Interceptions

   Use the following cout statements to ask the user for input:

```
cout<<"Enter pass completions "<<endl;
cout<<"Enter pass attempts "<<endl;
cout<<"Enter total yards "<<endl;
cout<<"Enter touchdowns "<<endl;
cout<<"Enter interceptions "<<endl;
```

   Each of your cout statements needs to be followed by a cin statement to get the user input.

   Your five inputs should then be used in the passer rating calculation as follows:

   vi. C = (completions per attempt – 0.30) * 5
   vii. Y = (yards per attempt – 3) * 0.25
   viii. T = touchdowns per attempt * 20
   ix. I = 2.375 – (Intercepts per attempt * 25)
   x. If C, Y, T, or I is less than 0, then set it to 0
   xi. If C, Y, T, or I is greater than 2.375, then set it to 2.375
   xii. PasserRating = (C+Y+T+I)/6*100

   Once you have the calculations working, add an evaluation of the passer rating. A rating is "poor" if it is 85 or below, "mediocre" if above 85, "good" if above 90, and "great" if above 95. Output the passer rating, and whether the rating is "poor", "mediocre", "good", or "great". For example, a print statement for a passer rating of 65 would look like: "*Rating 65, this is poor*".

   This cout statement should be something like:

```
cout<<"Rating "<<intRating<<", this is poor"<<endl;
```

To test your program, look up actual data on www.nfl.com or use the following information from 2007:

| Quarterback | Completions | Attempts | Yards | Touchdowns | Interceptions |
|---|---|---|---|---|---|
| D. McNabb | 180 | 316 | 2647 | 18 | 6 |
| T. Brady | 319 | 516 | 3529 | 24 | 12 |
| P. Manning | 362 | 557 | 4397 | 31 | 9 |

## Cycling: Power and Energy

*If you are interested in more information about these equations, or how they have been used in modeling cycling performance, there is a research paper posted on Moodle titled "The Mathematics of Breaking Away and Chasing in Cycling". It is provided purely for your enjoyment and background reading, you will not be tested on it.*

The power output for a cyclist (measured in Watts) can be calculated from the power needed to overcome air resistance and the power needed to overcome rolling resistance. *(Note: there are other variables that can be included, but for simplicity, we're just looking at these few.)* One set of equations to calculate power output are shown here:

$$P_{air} = k * CF_{draft} * v^3$$
$$P_{roll} = Cr * g * (M + M_b) * v$$
$$P_{\sec} = P_{air} + P_{roll}$$

where $P_{air}$ is the power to overcome air resistance, $P_{roll}$ is the power to overcome rolling resistance, and $P_{sec}$ is the total power output per second.

There are several other noteworthy parameters in these equations:

$M$ is mass of the rider in kg
$M_b$ is mass of the bike in kg
$v$ is velocity in m/s
$g$ is the gravitational coefficient ($9.8 \frac{m}{s^2}$)
$k$ is a coefficient related to rider position (0.18)
$Cr$ is a coefficient related to the resistance of the bike on the road (0.001)
$Cf_{draft}$ is a coefficient of drafting. Riders in a pack will be protected from the wind, and therefore require less power to overcome air resistance. The $Cf_{draft}$ value is generally between 0.80 and 0.60, representing a 20% to 40% reduction in power needed. Riders at the front of a pack have a $Cf_{draft}$ of 1.0, representing no reduction in power needed.

2. Write a program that takes as user inputs for $M, M_b, v, and\ Cf_{draft}$, and outputs the $P_{sec}$. To test your code, if you use *M = 63, $M_b$ =15, v = 10.8, $CF_{draft}$ = 0.70*, you should get *166.9 W*. Your output should be displayed as an integer, which would be 166W in this example, with no additional text in the display. Use the following cout statement:

   ```
   cout<<intPower<<"W"<<endl;
   ```

3. Once you have the power calculation working, add another user input to your code to ask for the distance in kilometers for the rider to travel. This input should be an integer and occur after you display the calculation result in Problem 2. Calculate the energy needed to travel that distance. This will require knowing the number of seconds required to travel that distance at the given velocity. You can get number of seconds needed to travel that distance using:

$$timeTravel = \frac{(distance * 1000)}{v}$$

   where *timeTravel* is time in seconds, *distance* is the distance in kilometers, and *v* is the velocity in m/s. To get total energy, use:

$$Energy_{total} = P_{sec} * timeTravel$$

   Print your result as an integer using the following cout statement:

   ```
   cout<<energyTotal<<endl;
   ```

## Loops, loops, loops

4. Write a block of code that creates an array of 5 integers. The code should then ask the user to input five numbers, one at a time. Once the array is populated, output the sum of the numbers at indices 0 and 1, 1 and 2, 2 and 3, 3 and 4. For example, if the 5 numbers stored in the array are [3, 21, 4, 20, 12], then your program should print:
   24
   25
   24
   32

   Use the following cout statement for each element in the array that you print:

   ```
   cout<<array[i]<<endl;
   ```

   *(This example assumes your array is called array and your loop iterator is i.)*

5. Using the same 5-integer array as in Problem 4, determine if the numbers are not increasing from the beginning of the array to the end of the array. If they are, print "True. Otherwise, print "False". For example, if you have an array called intArray then your program should print "True" if intArray[0] >= intArray[1] >= intArray[2] >= intArray[3] >= intArray[4].

   Use the following cout statement:

   ```
   cout<<"True"<<endl;
   ```

   or

   ```
   cout<<"False"<<endl;
   ```