3460:209 Assignment 5-A

# **Assignment 5-A: Sportsmania**

**Overview**

The purpose of this assignment is to make sure that you know how to write a program that contains functions and that does basic input, output, flow of control and/or calculations.

**PROGRAM SPECIFICATION**

For the assignment, we will write a program that simulates the answer to the greatest dodgeball player of all time while using functions.

In the land of Sportsmania, Aaron, Bob, and Charlie had an argument over which one of them was the greatest dodgeball player of all time. To end the argument once and for all, they agreed on a dodgeball duel. Aaron is a poor shooter and only hits his target with a probability of 1/3. Bob is a bit better and hits his target with a probability of 1/2. Charlie is an expert shooter and never misses. A hit means a part of another player was hit with the ball and that person hit drops out of the duel.

To compensate for the inequities in their marksmanship skills, it is decided that the contestants would fire in turns starting with Aaron, followed by Bob, and then by Charlie. The cycle would repeat until there was one man standing. And that man would be remembered as the greatest dodgeball player of all time.

a. Write a function to simulate a single shot. It should use the following declaration:

**void shoot(bool &targetAlive, double accuracy);**

This would simulate someone shooting at targetAlive with the given accuracy by generating a random number between 0 and 1. If the random number is less than accuracy, then the target is hit and targetAlive should be set to false. You need to generate random numbers.

For example, if Bob is shooting at Charlie, this could be invoked as:

**shoot(charlieAlive, 1.0);**

Here, charlieAlive is a Boolean variable that indicates if Charlie is alive. Test your function using a driver program before moving on to step b.

You may use constants to setup your inputs, for example:

**// Constants**

**const float AARONACCURACY = 1.0/3;**

**const float BOBACCURRACY = 0.5;**

**const float CHARLIEACCURACY = 1.0;**

**const int NUMDUELS = 1000;**

b. An obvious strategy is for each man to shoot at the most accurate shooter still alive on the grounds that this shooter is the deadliest and has the best chance of hitting back. Write a second function named startDuel that uses the shoot function to simulate an entire duel using this strategy (or code for the second strategy below). It should loop until only one contestant is left, invoking the shoot function with the proper target and probability of hitting the target according to who is shooting. The function should return a variable that indicates who won the duel. The prototype for startDuel is simple:

**int startDuel();**

c. In your main function, invoke the startDuel function 1,000 times in a loop, keeping track of how many times each contestant wins. A counterintuitive strategy is for Aaron to intentionally miss on his first shot. Thereafter, everyone uses the strategy of shooting at the most accurate shooter left in the game. This strategy means that Aaron is guaranteed to live past the first round, since Bob and Charlie will fire at each other.

Code the program to accommodate either (one or the other, exclusive) strategy and output the probability of winning for each contestant. The code to output the probability for strategy two, for example, from inside of main could possibly look like this (excluding variable naming):

**std::cout << "Using the strategy of shooting at the best shooter alive, " << std::endl;**

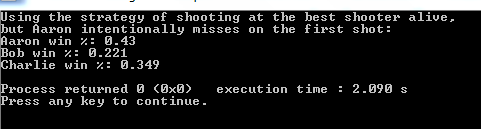
**std::cout << "but Aaron intentionally misses on the first shot:" << std::endl;**

**std::cout << "Aaron win %: " << (aaronWins/NUMDUELS) << std::endl;**

**std::cout << "Bob win %: " << (bobWins/NUMDUELS) << std::endl;**

**std::cout << "Charlie win %: " << (charlieWins/NUMDUELS) << std::endl;**

The output using the constants defined would look like this:



Remember to use one strategy or the other, but don’t do both. Make sure that your programs follow good documentation standards and follow the requirements for assignments. Reference the rubric standards on Brightspace. Note **functions and data validation are now required. Do not use using namespace std;.** By the way, you won’t need to validate these data as they are coded into your program.

Submission Instructions – for programming solutions

On Brightspace, go to the matching Assignments for the **ASSGN@-#**, where @ is the chapter and # is the number or character of the problem assigned (eg., 5-11 for chapter 5, problem 11), and submit the program (cpp) and any (hpp) files.

*Last updated 5.22.2016 by Will Crissey.*

*Be aware that programming falls under all of the rules of plagiarism. Be careful when using any coding found in the outside world that is not your own. Any evidence of plagiarism is subject to sanctions like forfeits, suspension, and even ejection, as determined by the Department of Student Conduct and Community Standards.*

**COPYRIGHT © 2017 Will Crissey, Jr., Assistant Professor, All rights reserved.**

This document is intended exclusively for use by The University of Akron's Department of Computer Science. Do not distribute document copies to the public in any way.

***THIS PROGRAM'S CONTENT IS PROTECTED***

The document and associated content are either the property of, or used with permission by, The University of Akron, Department of Computer Science and may be protected by copyright and other restrictions. Copyrights in the site content may also be owned by individuals and entities other than, and in addition to, The University of Akron, Department of Computer Science. The use of this document by you, or anyone else authorized by you, is prohibited, except for purposes of completing the homework/project. Any unauthorized use of the document may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.