COMP2710: Homework 3

Points Possible: 100
Due 16:59 March 29th 2019

Goals:

- To learn "while" and "do-while" statements
- To learn how to define functions
- To write a test driver
- To learn how to use assert()
- To use random numbers

Description:

In the land of Puzzlevania, Aaron, Bob, and Charlie had an argument over which one of them was the greatest puzzle-solver of all time. To end the argument once and for all, they agreed on a duel to the death (this makes sense?). Aaron was a poor shooter and only hit this target with a probability of 1/3. Bob was a bit better and hit his target with a probability of 1/2. Charlie was an expert marksman and never missed. A hit means a kill and the person hit drops out of the duel.

To compensate for the inequities in their marksmanship skills, the three decided that they would fire in turns, starting with Aaron, followed by Bob, and then by Charlie. The cycle would repeat until there was one man standing. That man would be remembered for all time as the Greatest Puzzle-Solver of All Time.

An obvious and reasonable 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 program to simulate the duel using this strategy. Your program should use random numbers and the probabilities given in the problem to determine whether a shooter hits his target. You will likely want to create multiple functions to complete the problem. My solution had only one function to simulate the duels and it passed in the odds and the three guys as pass-by-reference parameters. Once you can simulate a duel, add a loop to your program that simulates 10,000 duels. Count the number of times that each contestant wins and print the probability of winning for each contestant (e.g., for Aaron your might output "Aaron won 3612/10000 duels or 36.12%).

Strategy 2: An alternative strategy for Araon is to intentionally miss when both Bob and Charlie are still alive. Write a function to simulate Strategy 2. Your program will determine which strategy is better for Aaron.

Example I/O

The calculation results of your program will probobady be a little bit different from the provided output below due to the randomness in the program. But if your program tells you that strategy 1 is better then it is highly likely that there is something wrong in your code.

```
*** Welcome to the Duel Simulator ***
Unit Testing 1: Function - at_least_two_alive()
        Case 1: Aaron alive, Bob alive, Charlie alive
        Case passed ...
        Case 2: Aaron dead, Bob alive, Charlie alive
        Case passed ...
        Case 3: Aaron alive, Bob dead, Charlie alive
        Case passed ...
        Case 4: Aaron alive, Bob alive, Charlie dead
        Case passed ...
        Case 5: Aaron dead, Bob dead, Charlie alive
        Case passed ...
        Case 6: Aaron dead, Bob alive, Charlie dead
        Case passed ...
        Case 7: Aaron alive, Bob dead, Charlie dead
        Case passed ...
        Case 8: Aaron dead, Bob dead, Charlie dead
        Case passed ...
Press Enter to continue...
Ready to test strategy 1 (run 10000 times):
Press Enter to continue...
Aaron won 3751/10000 duels or 37.51%
Bob won 4139/10000 duels or 41.39%
Charlie won 2110/10000 duels or 21.1%

Ready to test strategy 2 (run 10000 times):
Press Enter to continue...
Aaron won 4209/10000 duels or 42.09%
Bob won 2520/10000 duels or 25.2%
Charlie won 3271/10000 duels or 32.71%

Strategy 2 is better than strategy 1.
```

Requirements:

1. You must follow the user interface provided above while implementing your program.

2. You must implement the following functions. Please use the provided function prototypes. Do not change the anything.

```
1) bool at_least_two_alive(bool A_alive, bool B_alive,
   C_alive)
   /* Input: A_alive indicates whether Aaron is alive */
   /*        B_alive indicates whether Bob is alive */
   /*        C_alive indicates whether Charlie is alive */
   /* Return: true if at least two are alive */
   /*         otherwise return false */

2) void Aaron_shoots1(bool& B_alive, bool& C_alive)
   /* Strategy 1: Use call by reference
    * Input: B_alive indicates whether Bob alive or dead
    *     C_alive indicates whether Charlie is alive or dead
    * Return: Change B_alive into false if Bob is killed.
    *         Change C_alive into false if Charlie is killed.
   */

3) void Bob_shoots(bool& A_alive, bool& C_alive)
   /* Call by reference
    * Input: A_alive indicates if Aaron is alive or dead
    *     C_alive indicates whether Charlie is alive or dead
    * Return: Change A_alive into false if Aaron is killed.
    *         Change C_alive into false if Charlie is killed.
   */

4) void Charlie_shoots(bool& A_alive, bool& B_alive)
   /* Call by reference
    * Input: A_alive indicates if Aaron is alive or dead
    *     B_alive indicates whether Bob is alive or dead
    * Return: Change A_alive into false if Aaron is killed.
    *         Change B_alive into false if Bob is killed.
   */

5) void Aaron_shoots2(bool& B_alive, bool& C_alive)
   /* Strategy 2: Use call by reference
    * Input: B_alive indicates whether Bob alive or dead
    *     C_alive indicates whether Charlie is alive or dead
    * Return: Change B_alive into false if Bob is killed.
    *         Change C_alive into false if Charlie is killed.
   */
```

3. You must implement a unit test drivers to test the function **at_least_two_alive**

4. You must use assert in your test driver.

5. You must define the hit ratios for the three puzzle-solvers and the total number of duels simulated as constant

<u>Hints:</u>

1. How to implement `“Press any Enter to continue...”`

   ```
   cout << "Press Enter to continue...";
   cin.get(); //Pause Command for Linux Terminal
   ```

   Note: you can implement the above two lines as a function to be repeatedly called by other functions in your program.

2. You may need to use the following libraries.

   ```
   # include <iostream>
   # include <stdlib.h>
   # include <assert.h>
   # include<ctime>
   ```

3. Initialize your random number generator as below:

   ```
         srand(time(0));
   ```

4. A sample code of generating a random number is given below:

```
/* Assume that the probabilty of hit a target is 25 percent */
int shoot_target_result;

shoot_target_result = rand()%100;

 if (shoot_target_result <= 25) {
    cout<<"Hit the target\n";

   /* add more code here */
 }
```

5. Please follow the following sample test driver to implement the test driver.

```
void test_at_least_two_alive(void) {
   cout << "Unit Testing 1: Function – at_least_two_alive()\n";

   cout << "Case 1: Aaron alive, Bob alive, Charlie alive\n";
   assert(true == at_least_two_alive(true, true, true));
   cout << "Case passed ...\n";
```

```
     cout << "Case 2: Aaron dead, Bob alive, Charlie alive\n";
     assert(true == at_least_two_alive(false, true, true));
    cout << "Case passed ...\n";

     cout << "Case 3: Aaron alive, Bob dead, Charlie alive\n";
     assert(true == at_least_two_alive(true, false, true));
     cout << "Case passed ...\n";

     /* add test cases 4-6 below */
     ...
}
```

Programming Environment:

Write a program in C++. Compile and run it with g++ (-std=c++11 is recommended) compiler on the tux machine, i.e., the AU server.

It is absolutely your responsibility to ensure your **code compiles and runs as intended on AU servers**. It is fine to use any IDE but please double check your program on a school machine.

Grading:

1. (5 points) Use comments to provide a heading at the top of your code containing your name, Auburn Userid, filename, and how to compile your code. Also describe any help or sources that you used (as per the syllabus).
2. (5 points) Your source code file should be named as "hw3.cpp". **Note:** You will not lose any point, if Canvas automatically changes your file name (e.g., hw3-2.cpp) due to your resubmissions.
3. (20 points) You must follow the specified user interface.
4. (5 points) Defined the required constants
5. (25 points) Implement five functions.
6. (20 points) Implement test driver for the at_least_two_alive
7. (5 points) You must use assert() in one of your test driver.
8. (5 points) Your program must compare strategy 1 with strategy 2.
9. (10 points) Quality of your source code.

Note: You will lose **at least 40 points** if there are compilation errors or warning messages when we compile your source code. You will **lose points** if you do not: (1)use the specific program file name, (2) have comments, (3) have a comment block on **EVERY** program you hand in.

<u>Deliverables:</u>

- Submit your cpp file on Canvas. The cpp file should be named as "hw3_username.cpp" where the username is your Auburn user ID, I.e. hw2_zzz0092.cpp
- Do **NOT** submit anything else!
- Do **NOT** submit zipped file. Submit your cpp file directly.

<u>Late Submission Penalty:</u>

- Late submissions will not be accepted and will result in a **ZERO** without valid excuses, in which case you should talk to Dr. Li to explain your situation.
-
- GTA/Instructor will not accept any late submission caused by internet latency.

<u>Rebuttal period:</u>

- You will be given a period of 3 business days to argue for your grade.