CS 10 - Assignment 5

Collaboration Policy

Collaboration between students on programming assignments is **NOT** allowed under any circumstances - you may not even discuss your work with others; you may **NOT** get coding assistance, or copy code, from **ANY** outside source *(books, web sites, current or past students, your own previous submissions if you are repeating the course, etc.)*. We test every submission for copying and when we find it we treat it as flagrant academic dishonesty on the part of ***all*** parties involved, regardless of their roles as provider or consumer. The penalty is generally instant failure in the course, in addition to the usual sanctions applied by Student Judicial Affairs.

As you can see from the submission header below, we ask you to essentially take an oath that the code you submit is ***entirely your own work*****.**

At the same time, we certainly understand very well that you will frequently need help completing your programming assignment, so we provide channels where you may get that help in a way which will strengthen your programming skills: instructor and TA office hours, and the discussion forums where you can ask all the questions you want about the assignment, and even help others with the understanding that you have gained *(but not, of course, with any actual code)*.

Assignment Submission Instructions

Your assignment must be submitted as a ***simple text file*** named ***main.cpp***
Files in ANY other format (MS Word document, etc.) or with ANY other name (main, main.doc, main.txt, etc.) will ***not*** be graded.

Submit your work via the appropriate iLearn assignment link, making sure you use the correct link and **click on the attach button**

We strongly recommend that:

1. you submit at least *6 hours* before the deadline, even if you haven't completed the program - you can re-submit as often as you like, and we will only grade the final submission
2. once you have submitted your final attempt, go back and download it back to your system: then run it just to make absolutely sure that it really is the file you intended to submit!

You are budding professionals, and are expected to take full responsibility for abiding by the requirements of a contract.

The ***only reason we will ever accept*** for a missed deadline is if the system administrators inform me that either the iLearn system or the CS department servers were off-line for a significant period around the time of the deadline *(In which case we will probably have notified you of alternative procedures)*.

Remember to include the following header information at the top of your program
(DO NOT REMOVE THE // at the beginning of each line, these tell the compiler to ignore these lines)

```
// Course: CS 10 <quarter & year>
//
// First Name:
// Last Name:
// Course username: <enter the username you use to login in the lab>
// Email address: <enter your cs or UCR student email address here>
//
// Lecture Section: <e.g. 001>
// Lab Section: <e.g. 021>
// TA:
//
// Assignment: <assn1, hw2, lab3, etc.>
//
// I hereby certify that the code in this file
// is ENTIRELY my own original work.
//
// =================================================================
```

NOTE: This header MUST appear at the top of EVERY file submitted as part of your assignment
(don't forget to fill in *your* details and remove the <> brackets!!).

Copy & paste this header into your file then update personal details.

Assignment Specifications

We are going to use the Monte Carlo Method to determine the probability of scoring outcomes of a **single turn** in a game called Pig.

Your Assignment

For this assignment you will simulate a given number of **hold-at-20 turns**, and report the estimated probabilities of the possible scoring outcomes. You are **NOT** implementing the game of Pig, only a single computer turn of this game.

What is Pig?

Pig is a folk jeopardy dice game with simple rules: Two players race to reach 100 points. Each turn, a player repeatedly rolls a die until either a 1 ("pig") is rolled or the player holds and scores the sum of the rolls (i.e. the turn total). At any time during a player's turn, the player is faced with two decisions:

- **roll** - if the player rolls
  - **1**: the player scores nothing and it becomes the opponents turn
  - **2 - 6**: the number is added to the turn total and the turn continues
- **hold** - The turn total is added to the player's total score and it becomes the opponent's turn.

Hold-at-20 Turn Strategy

Again, you are only implementing one turn of this game and then simulating it many times to estimate the probability of each scoring outcome of this turn strategy.

When implementing a single turn for the computer, one strategy to use is something called the **hold-at-20 strategy**. The hold-at-20 strategy has the computer continue to roll the die until it rolls a pig (1) which gives them a turn score of 0 or until it accumulates a turn score of 20 or more. In other words, the computer stops if it reaches a turn score of 0, 20, 21, 22, 23, 24, or 25.

For example, on the computer's turn, it rolls the following:

```
Roll 1: 2
Roll 2: 5
Roll 3: 6
Roll 4: 2
Roll 5: 4
Roll 6: 3
```

At this point the computer stops rolling since it has a score of 20 or higher (22 in this case). It didn't stop after `Roll 5` because its turn score was only 19 (2 + 5 + 6 + 2 + 4) at that point.

Random Seed Requirement

For this assignment, you **MUST** seed the random function with the value 1000 (instead of time(NULL)) so that you get these EXACT values (when compiled and run on CS server).

Input Requirements

Enter a single positive integer indicating the number of turns to be simulated.
(Larger numbers will tend to yield better estimations as we saw in lecture.)

Output Requirements

- Initially, prompt the user with: "`Hold-at-20 turn simulations? `"
- Output a blank line between the input prompt and table output.
- On the next line, print "`Score`" and "`Estimated Probability`" separated by a tab ('\t').
- After the simulations, print a table line for each score outcome that occurred in increasing order of score. **For each score outcome, print the score, a tab, and the fraction of turn simulations that yielded that score rounded to six digits after the decimal place.**

Example Runs

`USER INPUT IS` **`BOLDED`** `AND UNDERLINED.`

```
[user@well ~]$ g++ main.cpp
[user@well ~]$ ./a.out
Hold-at-20 turn simulations? 10000000


Score   Estimated Probability
0       0.624896
20      0.099483
21      0.095041
22      0.074062
23      0.054203
24      0.035110
25      0.017204
[user@well ~]$
[user@well ~]$ ./a.out
Hold-at-20 turn simulations? 1


Score   Estimated Probability
0       0.000000
20      0.000000
21      1.000000
22      0.000000
23      0.000000
24      0.000000
25      0.000000
[user@well ~]$
```

```
[user@well ~]$ ./a.out < input.txt
Hold-at-20 turn simulations?

Score   Estimated Probability
0       0.624896
20      0.099483
21      0.095041
22      0.074062
23      0.054203
24      0.035110
25      0.017204
[user@well ~]$
```

Notice that the results all have six digits to the right of the decimal. Also note, the 1 count simulation example is not meant to always produce `1.000000` in the `score 21` category but it will exist in some category and all 6 others will be `0.000000`.

Marking Guidelines

- Make sure your code compiles.
- Make sure your output score values, results and spacing match what is specified.
- We utilize the `diff` command to grade so make sure your spelling is correct.

Basic Style Guidelines

- **indentation** - follow proper 2-4 space indentation as you go into loops or branches
- **spacing** - blank lines should be used to separate logic blocks
- **no line wraps** - no line of code should have more than 80 characters
- "**meaningful**" variable **names**
- **comments** - each logical block of code should have a brief explanatory comment