# CS 171 Computer Programming 1

Winter 2017

Assignment 4: Quiz Show Strategy

## Purpose:

After completing this assignment you will have designed, implemented, debugged, compiled and provided documentation for a C++ program that involves loops and random numbers to do a simulation.

## The Assignment:

This assignment consists of:

1. An Introduction
2. Creating two functions that you will use in your final program.
3. A final program that uses the previously created and tested problems.
4. A document showing your analysis of the results.

In addition to the problem specifications, this document contains:

- *What to Submit*
- *Grading Rubric*
- *Academic Honesty*

Make sure you read everything!

---

## Introduction

There used to be a TV game show called "Let's Make a Deal" which ran for many years, with host Monty Hall.



www.letsmakeadeal.com

A typical situation in the game show was this: the stage is set up with three large doors labeled #1, #2, and #3. Behind two of the doors is a goat. Behind the other door is a new car.

You don't know which door has the car, but Monty Hall wants you to pick a door. Suppose you pick door # 1. Monty Hall then opens one of the other doors (say, door # 3) and shows you that there's a goat behind it. He then says: "I'll give you a chance to change your mind. Do you want to change your mind and pick door number 2?" After you make a decision, they open the door you finally picked and you find out whether you've won a new car or a new goat. To clarify: the locations of the car and the goats are fixed before your game starts. Monty does not get a chance to switch things around in the middle of the game. He just shows you that one of the choices you didn't make had a goat.

https://bueschermath.wordpress.com/2013/09/04/3/

There was considerable debate among fans of the show (as well as statisticians and mathematicians) about the following question: When Monty shows you that there's a goat behind one of the doors that you didn't pick, should you switch your choice? Or should you stand pat with your original decision? You will write a program to simulate this game and analyze this situation

Your program should have a loop that runs 10000 times. In each iteration, randomly pick a door number between 1 and 3 for placing the car. Place goats behind the other doors. Simulate the player with a random door pick. Randomly pick a door not the player's choice that has a goat (there may be one or two of them depending on what the player picked), simulating Monty's choice. Increment a counter for strategy 1 if the player wins by switching from their choice to the other door that is still unopened after Monty's reveal of a goat. Increment a counter for strategy 2 if the player wins by sticking with the original choice. When the loop is finished, display the percentage of times the user wins by using each strategy along with a message that answers the question: "is it to the player's advantage to switch doors?"

## Problem 1

Build a function with the prototype

```
void setupDoors(char &door1, char &door2, char &door3)
```

that randomly assigns the value `C' or `G' to the three reference parameters.

Test this function to check the following

1. Each call must check that two of door1, door2, and door3 is a `G', and that the other is a `C'.
2. You can trust the built in random library to do good enough at assigning the result fairly.

## Problem 2

Write a function with prototype:

```
void pickDoorChoices(char door1, char door2, char door3, int
      &doorPlayer, int &doorMonty)
```

This procedure should randomly set a value 1 through 3 for `doorPlayer`. The value of `doorMonty` should be set to an integer 1–3 that obeys the constraint that Monty never picks the same door as the player and never picks the door that has the car.

## Problem 3

After you know your functions work, complete the design and implementation of the experimental program that answers the question "is it to the player's advantage to switch doors after Monty tells them?" You should identify the need for, and then design and implement, other helper functions besides the two described in Problems 1 and 2

Your program should conduct 10000 experiments. Output a table of percentages indicating how often each strategy wins. Do other sets of 10000 experiments using several different values of the random number seed. From those experiments draw your conclusions.

## Problem 4

Create a PDF document using a word processor that includes:

1. Output from the several runs you made.
2. Appropriate tables and charts for your runs.
3. Your explanation/justification for your conclusions based on the experimental data.

## Submission

All homework for this course must be submitted electronically using Blackboard Learn. ***Do not e-mail your assignment to a TA or Instructor!*** If you are having difficulty with your Blackboard Learn account, ***you*** are responsible for resolving these problems with a TA, an Instructor, or someone from IRT, before the assignment it due. It is suggested you complete your work early so that a TA can help you if you have difficulty with this process.

For this assignment, you must submit:

Problems 1-2: NOTHING (kind-of)!!

- We will make sure your functions are used in part 3.

Problem 3:

- Your C++ source code file that contains all the functions of Programs 1-3. *Just submit your ".cpp" file, NOT your .dsp, .dsw, or similar IDE project files.*
- A PDF document with your User Manual
- A PDF document with your System Manual
- You **NO NOT** need to submit a test log for this assignment.

Problem 4:

- A PDF document showing the output of several runs, a table of those results and at least a paragraph on your conclusions based on the results.

## Grading Rubric:

1. Problem 1 Function Correct 20pts
2. Problem 2 Function Correct 20pts
3. Problem 3 doesn't have an syntax errors 10pts
4. Problem 3 produces correct result (proper logic) 25pts
5. Problem 4 shows results and reasonable analysis/conclusions 10pts

In addition, for Problem 3 (which includes Problems 1-3)

1. Coding style: consistent adherence to coding standards described in assignment 2 (the previous assignment). Through use of variable names, indentation, and comments it makes itself easy to comprehend. Those who want to adopt variations need justify their divergences and must provide in your written comments a justification that is persuasive and convincing to the grader. (You may wish to talk to the staff about this rather than assuming that you know what they like.) Coding styles reflect what code readers want to read, not just what programmers want to write.

    - 5 points = excellent. Code is included in a .cpp file submitted to blackboard, as well in readable form in the .pdf.
    - 4 points = good but a few improvements are in order
    - 2 points = many improvements needed for code to be readable for a general audience of programmers, or the formatting of the code in the .pdf or .cpp makes it unpleasant and difficult to read (grader's opinion is what matters here - if you have any doubts, you should ask before submitting.).
    - 1 point = missing major principles, should talk to staff about this to do better next time. This is the maximum number of points you can get in this category regardless of style quality if your program does not effectively print out the table it was designed to do.

2. Program construction

    - 5 points = excellent. Code is not needlessly complex. Functionality is encapsulate d in the mandated functions and used in that fashion in the program submitted. The program is written in a way that makes extension or changes easy to do. Global constants are used as mandated. Computations should not be needlessly complex, use math to simplify calculational expressions or logic to simplify if statements or complicated Boolean expressions. The grader's judgment is the relevant measure here.
    - 4 points = good, but a few improvements are in order.
    - 2 points = many improvements needed for code to come up to expectation for coding quality.

- 1 point = many flaws, should talk to staff about this. This is the maximum number of points you can get in this category if your program does not effectively print out the table it was designed to do.

3. External Documentation

   This includes content in the User Manual and System Manua (no Test Log this time).
   - 5 points = function definitions are commented with full explanation of purpose, parameters, results, and side effects, in the style of section 4.3 of the textbook "Function Comments".
   - 4 points = good, but some significant missing information that would ordinarily be expected
   - 2 points = needs significant improvement. Major miss.
   - 1 point = see the instructional staff to do better next time.

## Academic Honesty

You must compose all written material yourself, including answers to book questions. All material taken from outside sources must be appropriately cited. If you need assistance with this aspect of the assignment, see a consultant during consulting hours.

To ensure that assignments are done independently, in addition to human observation, we will be running all assignments through a plagiarism detection system. This program uses compiler techniques which are invariant of syntax and style. It has a very high accuracy rate.