CSC 112 Lab 2
Oh Craps

Spring 2015

Due
3:00pm Friday, 2/6

1 Introduction

The dice game craps first appeared North America in the early 1700's. The game was first played on riverboats in the United States and then moved west with the frontier as the country grew. There are two types of craps played today: *street craps* and *bank craps*. Betting is simplified in street craps, and someone must cover the bet in order for the game to progress. In contrast, bank craps is the variety played in casinos. The betting is more complicated and the house covers the bets (banks the game) so the players are playing against the house. We will play a modified version of street craps that does not include betting. The game consists one player rolling a pair of dice according to the following rules

1. Player rolls the dice.
2. If the sum of the dice (sum of the die faces) is 7 or 11 the player **wins**.
3. If the sum of the dice is 2, 3, or 12 the player **loses**.
4. Otherwise (player did not roll a 2, 3, 7, 11, or 12) the sum of the dice is called the *point*.
5. Player continues to roll until a 7 or the point appears. If player rolls the point, they **win**; otherwise, if the player rolls a 7, they **lose**.

1.1 The Odds of Craps

We are interested in determining the odds of winning in the game of craps (rules described above). A simple method for determining the odds of winning is to play several games and record the number of wins. The odds of winning would be the number of wins divided by the total number of games played. The more games that are played, the better the estimation.

The program will prompt the user for the number of games to play using a dialog window. After this number is entered, your program will then play the number of games requested. For each game, display in the console window: the game number, the result (win or lose), and the number of rolls required. At the end of the games, display the total number of games, the number won, the number lost, and the win percentage. An example for 5 games is given below.

Terminal

```
> ./a.out
Enter the number of games to play -> 5
Game 1 win, number of rolls 1
Game 2 lose, number of rolls 8
Game 3 lose, number of rolls 5
Game 4 lose, number of rolls 1
Game 5 win, number of rolls 1
Out of 5 games, won 2 and lost 3
Win percentage is 0.4
```

2 Modular Design and Unit Testing

Modular programming is a popular software design that emphasizes separating the functionality of a program into independent modules (functions in C/C++), such that each function contains everything necessary to execute only one aspect of the desired application. Modules represent a separation of concerns, and can improve maintainability by enforcing logical boundaries between components.[1] As result of this programming style, the `main` function becomes easier to read, since it contains only the general program flow and no program details.

Modular programming can also help program development and debugging, since each module has a clearly defined purpose or objective. When testing is done on individual *units* of code (functions in our case), then this is referred to as *unit testing*. Using this development technique, the programmer will create one function at a time, then test it (perhaps using a separate program) to ensure correctness before writing the next function. The idea is that if modules are well tested and shown to be correct individually, then assembling and integrating them into a larger program will require less effort.

2.1 Programming Requirements for this Assignment

The remaining programs for this course will follow modular program design and unit testing, including this programming assignment. For this assignment, your program must at least consist of the following functions.

- `inputNumberOfGames` is a function that will prompt the user and read the number of games to play.
- `playGame` is a function that will play one complete game of craps. The function should return if the player won or lost, as well as the number of rolls for that game.
- `outputGameResult` is a function that will display the result of an individual game (game number, result, and number of rolls).
- `outputGameStats` is a function that will display the win percentage for all the games.
- `rollDie` is a function that will simulate rolling a die. The following definition will simulate rolling a die by randomly generating an integer between 1 and 6 then returns the value using the function name.

  ```
  int rollDie()
  {  return (rand()%6 + 1);  }
  ```

  Where is the function `rand()` defined? It is defined in the `cstdlib` library, so you must add the compiler directive `#include <cstdlib>` to your program.

As a result, your `main` function will consist of variable declarations, function calls, and control structures (loops). No output or input should be done in the `main` function.

3 Programming Points

You **must** adhere to all of the following points to receive credit for this lab assignment.

1. Create a directory `Lab2` off of your `CSC112` directory to store your program in a file called `lab2.cpp`
2. Your program must be modular in design and consist of the five functions described in Section 2.1.
3. Your `main` function can only consist of variable declarations, function calls, and control structures (no input or output in the `main` function).
4. Your program must compile cleanly, no errors or warnings are allowed.
5. Pass variables appropriately. Only pass by reference when necessary.
6. Your program must adhere to documentation style and standards. Don't forget function headers and variable declarations.
7. **Turn-in** a print-out of your program source code (`lab2.cpp`). In addition, submit your program by copying it to your `Grade/Lab2` directory.

[1]See `http://en.wikipedia.org/wiki/Modular_programming` for details.