CS 222, Spring 2016

Project 1 - a simplified version of "Craps," a well-known dice game; Due Sat., March 12, 2016 by 11:59 p.m.

1. The basic rules of Craps are as follows:

a) If the player bets "for" him/herself:

- if the first roll of the dice results in a 7 or 11, then the player immediately wins the amount of the bet;
- if the first roll of the dice results in a 2, 3 or 12, then the player immediately loses the amount of the bet;
- if the first roll is a number other than 2, 3, 7, 11 or 12, the number that is rolled is called the player's "point."
  - The player may increase ("press") the amount of his/her bet at this point. For purposes of this game, the player should be allowed to double the amount of the bet.
  - At this point, the player's goal is to roll the same "point"again (called "made the point") before rolling a 7 (called "sevening out") for winning or losing the bet.

b) If the player bets "against" him/herself, the result is basically a mirror image of the above:

- if the first roll of the dice results in a 2, 3 or 12, then the player immediately wins the amount of the bet;
- if the first roll of the dice results in a 7 or 11, then the player immediately loses the amount of the bet;
- if the first roll is a number other than 2, 3, 7, 11 or 12, then the player who bets against him/herself is betting that s/he will roll a 7 before rolling the "point" second time.
  - If the 7 is rolled, the player wins; if the point is rolled first, the player loses.
  - As when betting for him/herself, the player should be allowed to double the amount of the bet if he/she wishes.

c) If the player won or lost, s/he must decide whether to quit or to play another game unless the player runs out of money.

d) Your program should give the player an initial betting billfold of $100.00. The minimum bet is $5.00. There is no maximum bet other than the amount of money available.

2. Specific design variables and functions

a) The main program and its variables

You will need to decide on appropriate variables in which to store the player's bank roll (in order to keep track of how much money the player has), how many times the player won and lost, and how many times the player bet for and against him/herself. Let's use an integer array to store how many times the player won, lost, bet-for, and bet-against .

This bank roll should be kept up to date on the player's current status, which means that wins should be added and losses should be subtracted from the bankroll as the user plays the game. After each game, the program must report the result of the game, the amount of money won or lost, the current value of the bank roll, how many times the player won and lost, and how many times the player bet for and against him/herself. After each game, the program should allow the player to continue playing until s/he chooses to quit, or until the player runs out of money. This central program control may be done within **main**().

b) "Rolling" the dice:

A separate **rolling**() function will be used to "roll" the dice with no input and three outputs. You will generate each dice value separately using a random number generator. Each random number generator needs to be seeded with the current time at the program execution. The possible values are one through six. This function will return the sum of the two dice values through function return value and the two dice values through address passing. This is to say that each dice value is actually stored in the variable of the calling funciton.

c) "Playing" the game:

A second function **playing**() will be used to play a single game of craps until the player either wins or loses a bet, based upon the rules given above. This function should be modify the current $ amount of the player's bank roll according to the game result, modify the array values of the player won or lost, and whether the player bet for or against him/herself. Within the function, the player is asked whether s/he would like to place a bet. If so, the player must choose whether to bet "for" or "against" him/herself (see game rules above). The player then "rolls the dice" (simulated by a call to the dice-rolling function rolling()). This should be done interactively (via a key press from the player), rather than simply having the program continuously roll the dice until the game is over. After each roll, this function should report the two random dice values and the sum of the two. If, after the first roll, the game is not over, the player should be asked whether s/he would like to double the amount of the bet. When a roll causes the end of a game, the player is notified whether s/he won or lost money in that game overall, and the amount won or lost. In all other cases, the player is notified that s/he needs to roll again.

d) "Ending" and "Beginning" of the game:

You need a separate function **ending**() to do the following: you should report the current value of the bank roll, how many times the player won and lost, and how many times the player bet for and against him/herself. You need to save the above information into a text file as well.

You need a separate function **beginning()** to do the following at the beginning of your program in main(): the function will open the text file you used to save game information for reading if it exists, so that your game can continue from previous played results. The function will initialize the bank roll, the times the player won or lost, and the times the player bet-for or against him/herself. If the file does not exist or the bank roll has no balance, you start the game from scratch as usual.

3. Additional functions may be necessary and/or appropriate, depending upon the final overall design of your program.

4. Testing and Submitting:

Test your program on Mason to make sure it compiles and runs properly in various types of cases.

When your program is finished, write a short program report in plain text or HTML format. Your report should comment on any special difficulties you encountered, and describe any bugs that still remain in your program. (Undisclosed bugs will be penalized more than disclosed bugs when your project is graded.)

While on Mason, create a scriptfile listing your program to the screen, compiling it, and demonstrating a sample interactive run where the user plays at least three full games. You should submit this script file, together with your project report and all source file(s) used for your project. (If your source code is contained in more than one file, please include a makefile with your submission as well.) Submit all files, including your project report and output text file to Blackboard.