Project 3: Casino

Overview: Your project is to create a simple casino simulator.

Goal:The goal of the project is to create and call your own functions and to practice test case development.

Description: The user can play any one of the games at the casino, for as long as they want and in any order that they want, provided they haven't gone bankrupt. The games at the casino are high-low, 21, craps and the slots. The user should be able to leave the casino (exit the program) at any time from the program's main menu.

Have the user start with a random amount of money between $100 and $200. The program should display a menu with the following options:

1) Play High-Low

2) Play 21

3) Play Craps

4) Play the Slots

5) Run Test Cases

6) Leave Casino

The program should have functions to perform each of the options 1-5 in the menu. Additionally, displaying the menu should be a function that is called from main( ) and you should create a function to calculate the amount won. A good modular design will require more than just these functions.

You should display the user's current balance before each game. After that, you should ask them how much they want to bet on that round of the game. Do not allow the player to bet more than they have or less than $1! You can assume that the user only bet whole dollar amounts.

Test Case Development: Your code should contain a function whose purpose is to test the code in your program. Option (5) above should call a “driver” function whose purpose is to validate the other functions. Here is the “pseudo-code” of what the driver should do:

Function testCaseDriver()

Validate bank balance

Update bank balance with a win

Update bank balance with a loss

Update bank balance with a negative number (should result in error message)

Prompt the user to determine how many times they want to test the Slots function

Loop through the input number of times

Call the play slots game with a random bet

You should make sure to comment exactly what the driver function is testing. Remember that you want to be sure to test:

* Common Cases
* Boundary Cases
* Error Cases

Your test cases will be graded on how thoroughly they test your regular code.

## Casino Game Descriptions

The following is a description of how each game at the casino is played:

* High-Low*:* The computer should randomly generate a number between 1 and 10, and display this number for the user. The user then picks whether the next number will be higher or lower than the first. The computer generates the next number and displays it. If the user was right, they win; if wrong, they lose! The user will always lose if the second number generated is the same as the first one. If the user wins, pay the betting amount (payoff = 1).
* 21*:* The object of this game is to get as close as you can to 21 points without going over. The user should start with two randomly generated values between 1 and 10 (we'll simplify the game by assuming aces always count as 1 point). To make this game work more realistically, you should randomly generate a value between 1 and 13, then make numbers bigger than 10 equal to 10 (this simulates Jacks, Queens, and Kings). Once the user's first two values are generated, display their sum. Ask the user if they want to hit or stay. If they hit, generate another value and add it to their sum. Keep giving them this prompt until they go over 21 (in which case they lose) or decide to stay. Now generate a random value for the computer between 16 and 23 and display it. If the computer gets a 22 or 23, it busts and the player wins. If the user has more points than the computer they also win, otherwise the computer wins. This game pays 2 times the betting amount (payoff = 2). HINT: It would simplify things to write a function that generates numbers for the user.
* Craps*:* Ask the user whether they want to bet on “Pass” or “No Pass”. The computer then generates TWO numbers between 1 and 6 (representing two dice being rolled) and displays them to the user.
  + If the sum of the two numbers is 2, 3 or 12 (“craps”):
    - and the user bet on “Pass” then the user loses automatically.
    - and the user bet on “No Pass” then the user wins automatically.
  + If the sum of the two numbers is 7 or 11 (“a natural”)
    - and the user bet on “Pass” then the user automatically wins.
    - and the user bet on “No Pass” then the user automatically loses.

If the sum is anything else (4, 5, 6, 8, 9, or 10) then the computer starts a loop and repeatedly generates new dice rolls until the new dice rolls come up the same sum as before or a sum of 7.

* + If the new dice roll sums to 7
    - and the user bet on “Pass”, then the user loses.
    - and the user bet on “No Pass”, then the user wins.
  + If the new dice roll matches the sum generated on the first turn,
    - and the user bet on “Pass”, then the user wins.
    - and the user bet on “No Pass”, then the user loses.

If the user wins then the game pays out the betting amount (payoff = 1).

* Slots*:* The computer should randomly generate three numbers between 0-9 and display them for the user to see. If all three of the numbers are the same, then the game pays 99 times the betting amount (payoff = 99). If just two of the three match, then the game pays 10 times the betting amount (payoff = 10).

After each game the program should return to the main menu, unless the user has gone bankrupt, in which case the program should display a farewell message and exit. Do not allow the user to keep playing after they have gone bust!

**Hint:** After winning: balance = balance + payoff\*amountBet.  
 After losing: balance = balance - amountBet.

**Hint 2:** When developing your project, you will want to use ***incremental development***, which is to say, you’ll want to write the simplest stand-alone function first, e.g. updateBalance(int& balance, int payoff, bool won). You’ll then want to add the code to test your updateBalance code (see pseudo code above). Once you’re confident updateBalance works, you can move on to the next function.

## Extra Credit:

Come up with another game for your Casino. Do not choose poker, as this is too complex to code in addition to the games listed above. But you can research other dice games, roulette, etc. If you add a game, make sure that you also add it to your test case function. It should behave similarly to the functions listed above – bet amount, calculation of win or lose, and payout amounts. **3-5 points per game – to get full credit it must be in testable**

Coding Requirements:

You **must** follow all of the good programming practices discussed in class:

* Comment your code thoroughly.
* Indent your code appropriately.
* Use meaningful variable names.
* No global variables.
* Provide the user with understandable prompts and instructions.
* Make sure your name is included in comments at the top of your code.
* You are NOT allowed to use goto statements in this or any other COMP 51 projects.
* …

If this is not done, points will be deducted from your program and it will be impossible to earn an ‘A’.

Submission: Using the Canvas assignment feature, you should submit the source code (.cpp file). **Make sure you submit your assignment after uploading the file attachment!**