Roulette Simulation

In this programming assignment you will be creating a limited simulation of roulette. Roulette is a game in which participants place wagers regarding the outcome of a ball landing in one of 38 numbered bins (0-36, plus an additional slot labelled 00) on a spinning wheel. You will simulate wagering, the ball landing on the spinning wheel, and the outcome of the wager; that is, whether the participant wins or loses the wager.

Starting the game: The better should start out playing with 1000 chips. Print out the possible wagers that users can make. For each possible wager, also list the payout for a winning wager. The table of possible wagers and payouts is below. Duplicate this table output in your own program.

```
-----------------------------------------------------
| Input | Wager                                | Payout |
-----------------------------------------------------
|   1   | Even numbers ( 2,  4,  6, ..., 36) | 1 to 1 |
|   2   | Odd numbers  ( 1,  3,  5, ..., 35) | 1 to 1 |
|   3   | 1st half     ( 1,  2,  3, ..., 18) | 1 to 1 |
|   4   | 2nd half     (19, 20, 21, ..., 36) | 1 to 1 |
|   5   | Exact guess  ( 1 - 36, 0, and 00 ) |35 to 1 |
-----------------------------------------------------
```

Input: The user should be prompted for their type of wager (**[1] Even**, **[2] Odd**, **[3] 1st half, [4] 2nd half,** or **[5] Exact Guess**). Implement this as a menu and incorporate into the printed table listed above. If the user wagers on an exact number, they should also be asked for the number on which they would like to wager. Finally, the user should be asked for an amount of chips to wager. However, they must wager a positive amount and may not wager more chips than they currently possess. If *any* of the player's inputs are invalid, notify them of the mistake they made and allow them to re-input until they enter a valid input.

Once validated, the entered information should be reflected back to the user and the user should be given a choice to take back their wager or proceed with the game. For example, suppose that the user wagered **100 chips** on **even**. Then a prompt like this should appear:

```
You have decided to wager 100 chips on even.

Is this correct?
Enter 'Y' to proceed with the game
OR enter 'N' to take back your wager:
```

If the user enters 'Y' or 'y', then the "spin" proceeds. If the user enters 'N' or any other input, then they forfeit their wager and they will not proceed with that roulette spin (they will have to wait until the next spin).

Simulating the spin: To simulate the ball landing in one of the 38 bins on given a spin (create an `int` called `winningNum`), you will employ C++ functions and other code that randomly selects an integer value between -1 and 36. To do so, use the random number technique described in class and printed in the lecture notes. Note that

the value -1 will represent 00 in code, but 00 should be what is displayed in the console (ask for clarification on this if you have any questions!).

*It is advised that you get the entire program working with a pre-selected value for `winningNum` before trying to get your program to run with a random number.

Winning/Losing Wager: Handle winning and losing according to the wager and payout schedule printed above. An example might help explain:

Suppose that the ball lands on 17, making 17 the winning number. If the bettor wagered 100 chips on an **odd** number, then the bettor is given 200 chips (the 100 chips are returned and an additional 100 chips are awarded). Handle likewise for if the better wagered on the ball landing in the **1st half** of valid numbers (1-18). If the bettor wagered on an **even** number, the wagered money is not returned since 17 is not an even number. Handle likewise for if the better wagered on a **2nd half** number, since 17 isn't in the second half of valid numbers (19-36).

If the bettor instead wagered 100 chips on an **exact 17**, then the bettor is given 3600 chips (the 100 chips are returned and an additional 3500 chips are won). *Note: although you may place an exact wager on 0 or 00, bettors wagering on even (2, 4, 6, …, 36) or first half (1-18) numbers do not get paid if the ball lands on 0 or 00.*

Output: The user should be informed of the number on which the ball landed. If the user has a winning bet, then their winnings should be shown according to the payout schedule given above. If the user has a losing bet, then they should be informed of the loss. Their running chip balance should also be given to them.

*Although there is no strict rule for how to format the output, a portion of the grade will be based on the neatness and readability of the console text (only console input and output are necessary).

Repetition: The above steps should be repeatable. After each "spin," prompt the player, asking if they would like to play again. If they respond with 'Y' or 'y', let them play again. Otherwise, end the game. However, only allow them to play again if they have more than 0 chips. If they don't, inform them and end the game.

Post-game: Once the game has ended. Inform the better of how many chips they walked away with. Furthermore, if they walked away with more chips than they started with, congratulate them. If they walked away with 0, print something consoling (e.g. "Tough break, kid") Finally, thank them for playing and end the program.

This project is worth **100** points and is **due on Wednesday, Feb. 2nd before class.** Submission must include:

- An electronic turn-in through Canvas. The entire Visual Studio *solution folder* must be zipped in order to do this. To create a zipped folder, right-click the folder and select **Send to -> Compressed (zipped) folder**. Rename this folder "**Project1*_firstname_lastname***" where *firstname* and *lastname* are replaced with your first and last name, respectively. Upload this zipped folder to the Project 1 assignment page.
- A paper printout of your code, along with screenshots of the console **from a variety of program runs** to illustrate the program's ability to correctly handle different inputs (for example, ***at least*** show validation of inputs, taking back a wager, a winning bid on an exact number, a winning bid on an odd/even number, a losing bid, looping until the user quits or is out of chips, etc.). Submit to Dr. Ringenberg's office.

*Please paste all screenshots into a Microsoft Word document before printing.