## Aim of Exercise

The aim of the exercise is to see the way you approach creating a new program, the way you structure your code, and the way that you deal with any unexpected problems. For this exercise, these are given a higher priority than producing a complete and working solution. We would rather see an excellent but incomplete solution than a badly structured complete solution.

Please try to attempt each of the actions below. If you are stuck with an implementation detail of one action, move onto the next action. We would rather see an incomplete solution that covers all actions than a complete solution of one action, and no attempt at another.

This exercise is expected to take about 60 minutes. We suggest you spend some time initially to read thru the entire problem and plan a structure for the program before you begin. Though you may take more than 60 minutes, it is entirely up to you. Creating a completely functioning program is not required. We are more interested in seeing how you structure the program and code. This exercise will be timed.

## Specification

This program simulates a game of roulette. In roulette, a wheel spins and yields a number between 1 and 36 when a ball drops into a numbered slot. There are also slots numbered 0 and 00. These slots are green and each of the other slots is red or black. For the purposes of this program, **assume that all odd numbers are black** and **all even numbers are red**. In the real game, the colors black and red are distributed differently, but we will ignore that for the purposes of this program. Additionally, 0 and 00 are not considered either even or odd.

In Las Vegas style rules, gamblers can make several different kinds of bets, each of which pays off differently as listed in the table below. These payoffs are close to the real mathematical odds of getting such a spin, but slightly lower to allow the casino to make a profit on average.

| **Bet** | **Payoff** |
|---|---|
| Red/Black | 1 to 1 |
| Odd/Even | 1 to 1 |
| Single number | 35 to 1 |
| 1st 12 (1-12)/(13-24)/(25-36) | 2 to 1 |
| Low (1-18) / High (19-36) | 1 to 1 |

A payoff of 1 to 1 pays *even money*: a bet of $10.00 earns the gambler $10.00 if the bet wins. On the other hand, a payoff of 17 to 1 means that a bet of $10.00 earns the gambler $170.00. If the gambler loses, no money is paid out and the bet is kept by the casino.

***Input / Output***

Your program should prompt the user for an initial amount of money, the user's bankroll. The user should then be given the opportunity to make one of three different bets, or to quit the program. You are required to implement the first three bets in the table above; you can implement the last two bets for one point of extra credit each. Initially you should present the user with a menu similar to the list below:

- bet on red or black
- bet on 1st 12, 2nd 12, 3rd 12: (1-12)/(13-24)/(25-36)
- bet on a specific number
- bet on high/low

Then, for each bet, you should prompt the user for information specific to that bet. For example, if the user chooses to bet on a specific number, you should prompt the user to enter the number. It is fine to represent the number "00" as "-1".

The first two bets pay even money: a bet of $5.00 earns $5.00 if the bet wins. Winning a bet on a specific number pays 35 * b where b is the bet. Thus a winning bet of $4.00 earns $140.00. A winning bet earns back the money bet so that betting $10.00 on red and winning means that you get back $20.00 --- the original wager plus the even money winnings, or 10 + 10 = 20.

The user should be allowed to bet any positive amount not exceeding the bankroll. After each bet, the bankroll should be updated appropriately and the new total reported to the user. If the user runs out of money, the program should display a message saying so.

After user is done placing bets have a button to spin the wheel and see the resulting number along with the winnings added to the bank roll.