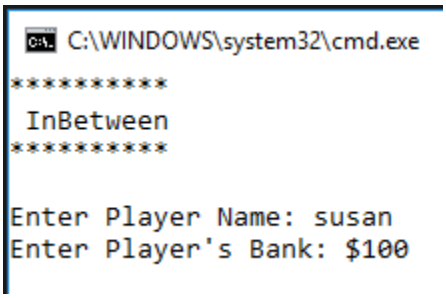
CS250 Assignment 5
In-Between: the card game

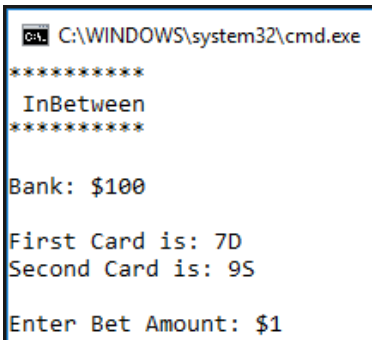
Date assigned: Wednesday, March 8, 2017
Date due: Friday, March 17, 2017
Points: 35

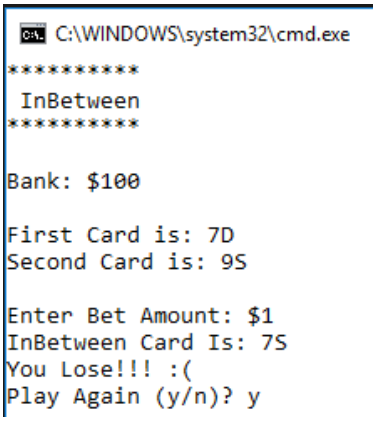
In-Between (AKA Acey Duecy) is a card game, typically played with multiple players, where two cards are displayed face up and players bet on whether the next card to be displayed falls in between the two cards. (http://en.wikipedia.org/wiki/Acey_Deucey_%28card_game%29).

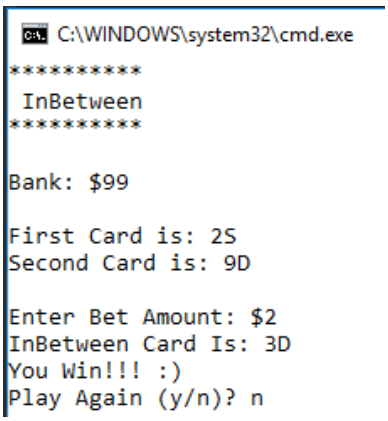
You are to implement a single player version of this game. Two cards will be displayed, and the user will place a bet on whether the next card will fall in between the two cards. If the third card does fall between the first two cards, then the user will win their bet ☺; otherwise, they lose their bet ☹ User input for the following example is susan, 100, 1, y, 2 and n. I used a different shuffling algorithm than described further on for the output below.

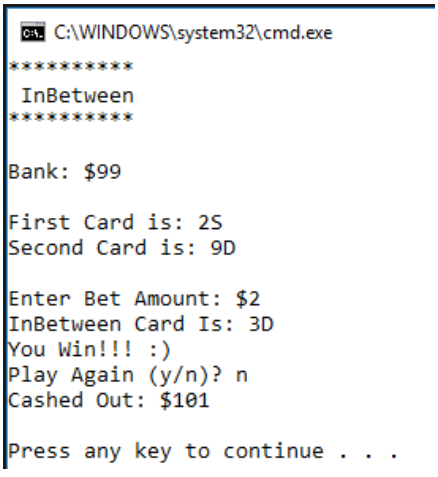
Example:











Notes on gameplay:

- The game begins with the player entering their name and buying into the game. The player must buy into the game with a value greater than 0. Continue asking for input until the player meets this requirement.
- Next, two cards are dealt and displayed as shown in the second screenshot. When displaying the cards, always display the lowest card first and the highest card second. The card values from lowest to highest are 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A.
- After both cards are displayed, the player places a bet. The minimum bet amount is $1 and the maximum is the floor of half of the balance unless the player has less than $10 in which case the player can bet everything. All bets are to be whole numbers and will be entered as such. Continue asking for input until the player meets this requirement.
- Once the bank balance for the player reaches zero, then the game is to display the message "Cashed Out: $0" and exit.
- I will not test more than a deck's worth of card, so there will no need to test for and reshuffle the deck.
- Let's assume that a player starts with a balance of $100.
- If the player bets $10 and wins, the resulting balance will be $110 as the player wins their bet.
- If the player bets $10 and loses, the resulting balance will be $90.
- Play continues until the player decides not to play again (cashes out) or goes broke. The player must enter a y or an n to cash out. Continue asking for input until the player meets this requirement.

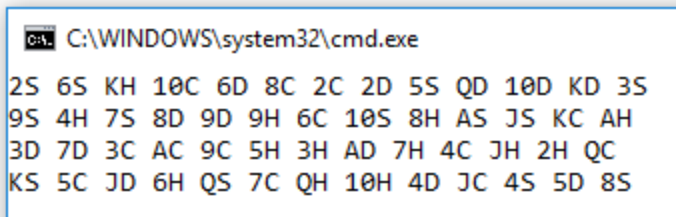
Notes on design:

Your object-oriented design must use at least 3 classes (Card, Deck, and InBetween). For each class you will need a .h and a .cpp. You will also need a driver that plays the game. Your project must have at least 7 files in total (Card.h, Card.cpp, Deck.h, Deck.cpp, InBetween.h, InBetween.cpp, and InBetweenGame.cpp). For this assignment, place all files in the project 04Card that you created for the last assignment. All .h files go in Header Files and all .cpp files go in Source Files. Finally, you must use the Card class created from the previous assignment and rename your driver as InBetweenGame.cpp.

Remember, cards are numbered from 0 through 51 (MAX_CARDS_IN_DECK – 1) inclusive for a total of 52 cards; therefore, initialize your deck of cards with the cardIds from 0 through 51 inclusive AND in that exact order. You must use the following shuffling algorithm assuming the random number generator is seeded with a seed value of 2 in the constructor of Deck:

1) Set a variable cardIndex equal to the position of the last card in the deck which is 51.
2) Generate a random number in the range of 0 to (cardIndex – 1) inclusive. For the first iteration of the loop, the random number will be between 0 and 50 inclusive, second iteration 0 through 49 inclusive, ... Place this random number in a variable called randomIndex.
3) Exchange the card at position cardIndex with the card at position randomIndex
4) Decrement cardIndex by 1
5) if cardIndex is not equal to 0, go to 2)

My shuffled card deck looks like the following. Your shuffled deck must match exactly to make sure we are using the same shuffling algorithm and seed when I go to grade.



Goals for this assignment

1. Write an object-oriented program using at least three classes.
2. Avoid using getters and setters when possible.
3. Understand the use of composition in OOP.
4. Practice modular programming by using well-defined functions.
5. Write 1 function prototype, implement that 1 function prototype, and test that 1 function BEFORE writing another prototype!!!!

To complete this assignment you must submit the following:

An electronic copy of your program on Grace

a) Add the above described files to your project called **04Card** in your existing solution **PUNetID-Assignments**. It is vital that you name your project correctly!

b) Type your program (fully documented/commented) into the project. You need to follow the coding standards from the CS250 Web page. These coding standards have been modified to include additional C++ language features introduced in CS250, so please be sure to read the new coding standards.

c) Pay attention to the example output. Your program's output must look **exactly** like the sample output. The spacing and newlines in your output must match exactly.

d) Make sure that your program builds without errors & warnings and runs correctly. If you get any errors or warnings, double check that you typed everything correctly. Be aware that C++ is case-sensitive. You will lose 10% if there are any warnings and 40% if your program does not build successfully.

e) Once you are sure that the program works, it is time to submit your program. You do this by logging on to Grace and placing your complete solution folder in the correct drop folder based on the section of the course in which you are enrolled (**CS250-XX Drop**).

f) The solution must be in the drop folder by the time class starts on the day the assignment is due. Anything submitted after that will be considered late.

g) If you drop multiple solutions, you will lose 10% of the assignment points, so do not drop until you are entirely sure you are completely done working on the assignment.

A hard copy of your program

a) The hard copy must be placed on the instructor's desk by the time class starts on the day that it is due.

b) The hard copy must be printed in color, double-sided, and stapled in the upper left corner if your solution contains multiple pages.

c) Your tab size must be set to 2 and you must not go past column 80 in your output.

Remember, if you have any problems, come to me straight away with your project on a flash drive or on Grace. Good Luck!!!!