**Scoring Rubric for Project 5: Blind Man’s Bluff**

*Due 10/24/2019 @ 3:30 pm*

|  |
| --- |
| Student Name: Sarah Ruth Nicholls |

|  |  |  |
| --- | --- | --- |
|  | **Score** | **Maximum** |
| **Execution (50 pts):** | | |
| Program compiles without errors (warnings are okay) | 50 | **50** |
| **Implementation (40 pts):** | | |
| Implements a default constructor and one alternate constructor and implements one member function that returns a string specifying the value of the card in the format of “Jack of Hearts” in Card class | 5 | **5** |
| Implements at least one overloaded comparison operator (such as < or >) for Card class following the given ordering rules | 2.5 | **5** |
| Implements copy constructor for Deck class that allocates memory and performs deep copy | 4 | **5** |
| Implements destructor for Deck class that frees memory | 5 | **5** |
| Implements assignment operator overload that frees old memory, allocates new memory, performs deep copy, and handles self-assignment case | 5 | **5** |
| Implements a member function that populates the deck with 52 cards. | 5 | **5** |
| Implements a member function that shuffles the cards in the deck by performing cardsLeft^2 swaps of two randomly chosen cards in the deck | 4 | **5** |
| Includes a Deck member function that removes a card from the deck and returns its value and implements a member function that adds a card to the deck (or returns false if there is no physical space in the deck). | 4 | **5** |
| Asks the user to play again after each turn; the game ends when the player chooses not to play or the deck is empty with the user’s number of wins and losses. | 5 | **5** |
| **Style (5 pts):** | | |
| The driver and functions are easy to follow based on the use of comments | 3 | **3** |
| Easily identifiable variable names | 2 | **2** |
| **Total (100 pts):** | 94.5 | **100** |

Notes:

Your overloaded comparison doesn’t take the suit of a card into account.

Your copy constructor should initialize arraySize before you use it in the for loop.

It’s good practice to include cards = nullptr; in your destructor so it won’t be a dangling pointer.

You are not increasing cardsLeft in your addCard function, so if you call the function twice in a row, you are just overwriting the same memory location.

In your shuffle function, you need either cardsLeft\*cardsLeft or pow(cardsLeft,2) instead of cardsLeft^2. The syntax of the last statement does not raise a number a power; it is actually a bitwise operator.