Lab 03 - Encapsulation

Instructions:

- **Playing cards contain suits and symbols. For the standard deck of cards, its valid suits are spades, hearts, clubs, and diamonds; and its valid symbols are ace, numbers 2 through 10, jack, queen, and king. While for an Uno deck of cards, its valid suits are red, yellow, blue, green, and black; and its symbols are the numbers 0 through 9, draw 2, reverse, skip, wild, and wild draw 4. Your objective is to create a *Card* class and define a couple of functions that create decks of *Card* objects.**
- **Your submissions must be submitted to the GitHub repository in the Lab03 directory.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the lab.**

Grading

Task	Name	Maximum Points	Points Earned
1		2	
2		1.5	
3		1.5	
Total		5	

Task 1

In a header file named `"Card.h"` within the namespace *LB3* define a class named *Card* that must contain

- a private char field named *suit*.
- a private char field named *symbol*.
- a public default constructor that assigns `'X'` to both fields.
- a public copy constructor.
- a public assignment operator.
- a public empty destructor.
- a public constant getter method for *suit* named `Suit()` .
- a public constant getter method for *symbol* named `Symbol()` .
- a public setter method for *suit* named `SetSuit()` that assigns the uppercase of the parameter to *suit*.
- a public setter method for *symbol* named `SetSymbol()` that assigns the uppercase of the parameter to *symbol*.
- a public string constant method named `ToString()` that takes no paramaters. It returns a string in the format

  $$[x:y]$$

  where x and y are the values of *symbol* and *suit* respectively.
- a friend overloaded ostream operator that returns its output in the same format as `ToString()`.

Task 2

In a header file named `"StandardDeck.h"` within the namespace *LB3* define a *Card* pointer function named `BuildStandardDeck()` that takes no parameters. It must return a dynamic array of *Card* objects that represent a standard playing cards deck. Use `'S'`, `'C'`, `'D'`, and `'H'` for the suits spades, clubs, diamonds, and hearts respectively. And use `'A'`, `'T'`, `'J'`, `'Q'`, and `'K'` for the symbols ace, ten, jack, queen, and king respectively, and for the other symbols use their respective digit. A standard deck of cards consists of 52 cards such that each symbol is paired with each suit.

Task 3

In a header file named `"UnoDeck.h"` within the namespace *LB3* define a *Card* pointer function named `BuildUnoDeck()` that takes no parameters. It must return a dynamic array of *Card* objects that represent an Uno deck. Use `'R'`, `'Y'`, `'B'`, `'G'` and `'K'` for the suits red, yellow, blue, green and black respectively. And use `'T'`, `'R'`, `'S'`, `'W'`, and `'F'` for the symbols draw 2, reverse, skip, wild, and wild draw 4 respectively, and for the other symbols use their respective digit. An Uno deck consists of 108 cards. Each color suit except for black contains 25 cards; one 0, and two sets of 1 - 9, draw 2, reverse, and skip. And there 4 each of wild and wild draw 4 all in the black color suit.

Extra Credit

In a header file named `"Extra.h"` define a class named *Password* that must contain

- a private string field named *value*.
- a public default constructor that assigns `"password"` to the field.
- a public copy constructor.
- a public assignment operator.
- a public empty destructor.
- a friend overloaded equal operator that takes two constant *Password* reference parameters and returns true only if their *value* fields are equal.
- a public setter method for *value* that assigns the parameter to *value* only if the parameter is at least 8 characters long with at least 1 uppercase letter, at least 1 lowercase letter and at least 1 digit.
- a public string constant method named `ToString()` that takes no paramaters. It returns a string of asterisks that is the same length as *value*.
- a friend overloaded ostream operator that returns an output in the same format as `ToString()`.

(2 points)