Assignment 08

Link:
https://docs.google.com/document/d/1eaBxYZtteNw_l4qeHGwwr7pJqOzU4uWblyTUDwLWEMI/edit?usp=sharing

Due: Monday, April 25th, 2022, at midnight, 100 points.

Documentation Header Reminder

Before you start your assignment, you will need to add documentation similar to what we demonstrated in the first few lectures.

Function Prototype Documentation Reminder

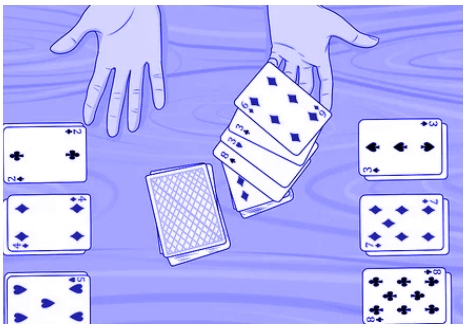
In the function prototype section of your C++ program, remember to add documentation to each function similar to what was demonstrated in lectures.

Separate File Compilation

For this assignment (and onward), you will submit multiple C++ compilable files containing a program written in C++. Name your file a meaningful name and give it a proper extension (.h, .hpp, .cpp). Also, make sure that you compile and run your program using the GNU (g++) compiler before submitting to make sure that it will work.

Card Wars

Background



You are the ruler of a very small country with incredibly odd laws. For the past four years you have had a very successful reign. However, one of the unchangeable laws of your country of *Fifteenseventystan* is that the position of ruler can be taken by anyone who beats the current ruler at a game of war. Today, you have been challenged by a young adventurer covered in slime, for some reason. In order to keep your position, you must beat him at war.

Modified Game Rules

War is a simple two-person card game. Each player receives half of the deck. Each player will then flip over one card at a time. Whoever has the higher valued card will then receive both cards, and place them at the bottom of their deck. If the values of both cards are the same, both cards will be destroyed and removed from the game forever. This will continue until a player has equal to or less than 10 cards remaining in their deck. When this happens, the other player will be announced the winner.

The card values are, in ascending order, 2-3-4-5-6-7-8-9-10-J-Q-K-A. Each card will also have a suit, either clubs, hearts, diamonds, or spades. Therefore, the full deck has a total of 52 cards. Specifications for how cards are ranked will be stated later in the assignment description.

Specifications

For this assignment, you are going to create several classes.

- ☐ The card class (used to store the value of a single playing card)
  - ☐ **Private members**:
    - ☐ A character to represent the **cards value** (2, 3, 4, 5, 6, 7, 8, 9, J, Q, K, or A)
    - ☐ A character to represent the **suit** (H for heart, C for club, S for spade, and D for diamond)
    - ☐ A boolean to represent whether or not the card has been **destroyed**.

  - ☐ **Public members**:
    - ☐ A **default constructor** should be made available that does *nothing*. This is simply put in place to guarantee a successful compilation of the program.
    - ☐ A **parameterized constructor** should be made available to take in both the card's value and its suit.
    - ☐ **Overloaded + operator**: The + operator should be overloaded within this class. When two cards are added together, it should return an integer which is the sum of the values of the 2 cards. For purposes of this function, count J,Q,K, and A as having a value of 10. For example, if you added the 7 of clubs and the queen of diamonds together, it should return a 17. If the sum of two cards is 11, both cards should be destroyed.
    - ☐ **Overloaded > and < operators**: These operators should be overloaded to compare the value of two cards. CardA>CardB should return true if card A is greater than card B, and false if otherwise. Similarly, CardA<CardB should return false if card A is greater than card B, and true if otherwise. How cards are compared are as follows:
      - ☐ If the two cards have *different suits*, then the card with the higher value is greater. The only exception to this is when the cards' values are 2 and A, in which case the value 2 will be considered the greater card.
      - ☐ If the two cards have the *same suit*, then the card with the lower value is greater. The only exception to this is when the cards' values are 2 and A, in which case the value A will be considered the greater card.

- ☐ **Overloaded == and != operators**: This operator should compare two cards. The == operator overload should return true if the two cards have the same value, and false if otherwise. Similarly, the != operator should return false if the two cards have the same value, and true if otherwise.
- ☐ **Overloaded ~ operator**: This operator should destroy the card forever.
- ☐ **Overloaded insertion operator (<<)**: This operator should return an ostream object by reference. It must output the card's value, followed by its suit. For example, the 7 of clubs would be output as 7C.

- ☐ The Deck class (used to represent a deck of cards)
  - ☐ **Public members**:
    - ☐ **A default constructor** should be made to populate the deck with all 52 cards.
    - ☐ A **parameterized constructor** should be made available to take in an array of cards, and an integer representing the size of the array of cards. The deck should be set equal to the card array.
    - ☐ A **shuffle** function: This function should randomize the position of all of the cards within the card array.
    - ☐ A **getCard** function: This function should take an integer parameter as the index. It should return the card within the card array at that index position.

  - ☐ **Private members**:
    - ☐ An array of at most 52 card objects.
    - ☐ An integer to represent how many cards are actually in the deck.

Overall Program Flow

1. Create a deck of 52 cards with your default constructor.
2. Shuffle the deck.
3. Create two arrays of cards, one which contains the first 26 cards in the deck you just created, and one which contains the remaining 26 cards in the deck you just created.
4. Create two more decks of cards, setting the cards they contain equal to the arrays you created in step 3.
5. Compare the cards in each of your 2 decks one by one, starting from the beginning of the cardArray, and ending at the end. When you hit the end of either player's card array, start again from the beginning of the card array.
6. If a card is destroyed, leave it in place in the array, and decrement the number of cards in that deck by 1.

7. If a card that has been selected has already been destroyed, simply move on to the next card in that player's array. Do not move onto the next card in the other players array unless it has also been destroyed.
8. This process should repeat until a player has 10 or fewer cards. When this happens, announce that the game is over, and announce the winner.

Notes

- Set the random seed for this assignment to be 469.
- It is recommended that each class definition goes in their own header file.
- You may define additional functions to be used to improve the program's organization. This includes global functions and member functions for the classes as well.
- You may use functions from existing libraries if 1) they are explicitly stated in the assignment or 2) the usage of the library functions has been thoroughly introduced in class. Otherwise, the usage of functions from existing libraries is prohibited without the permission of your instructor.
- If the parameters and return type of a function is not specified, then it is your responsibility to determine the most appropriate function signature.

Sample Outputs

Sample outputs regarding the format are provided below for your reference, i.e. the data values used in the output might be completely made up. Your actual program output does not have to match the sample exactly. Make sure that yours is nicely formatted and is readable to anyone.

Sample Output

This is war!
Input defender name:

Julian

Input attacker name:

Normo

Battle commence!

//Sample Output Coming Soon

–Game over–

Congratulations Normo, you have won! Enjoy your new country!