War

Take-home programming test instructions

Prerequisites

It is highly recommended to use **Visual Studio** to create your project. If you do not have visual studio, the community edition can be downloaded for free and is suitable for this exercise.

https://visualstudio.microsoft.com/vs/community/

Background

War is a card game played with a standard deck of 52 playing cards.

From wikipedia [https://en.wikipedia.org/wiki/War_(card_game)]:

The objective of the game is to win all of the cards.

The deck is divided evenly among the players, giving each a down stack. In unison, each player reveals the top card of their deck—this is a "battle"—and the player with the higher card takes both of the cards played and moves them to their stack. Aces are high, and suits are ignored.

If the two cards played are of equal value, then there is a "war". Both players place the next three cards face down and then another card face-up. The owner of the higher face-up card wins the war and adds all the cards on the table to the bottom of their deck. If the face-up cards are again equal, then the battle repeats with another set of face-down/up cards. This repeats until one player's face-up card is higher than their opponent's.

House Rules

War has a wide variety of house rules, for this exercise we will be using the following:

- Instead of placing cards at the bottom of a player's deck, place them in a discard pile. Once the deck is empty, the discard pile should be shuffled and used as the player's deck.
- If a player runs out of cards in a "war", they should use as many cards as possible. If there are further ties, their last placed card continues to be the card used for comparisons.

The Exercise

Create a console application to play War against a computer opponent using C# or C++. Confirm that the player wishes to play and then start a new game. Players should play cards by hitting enter and be able to see what they played, what the opponent played, and who won. Information such as the player and opponent deck/discard size should be presented or available. Any additional features are optional. This is a very open problem, and it's up to you to decide how you want to solve it. **Aim to take around 2-4 hours of time coding your solution, and do not spend more than 6 hours.**

What We're Looking For

- Your game should be playable from start to completion.
- You handle any possible game states.
- Your code is straightforward and easy to read and understand. Comments are recommended, especially around any potentially confusing aspects of your implementation.
- You've considered how to test your code.

Submission

Your submission must include the following:

- Your source files including any sln or proj files. Do not include unnecessary files.
- A rough breakdown of the time you spent implementing the exercise. Please explain how your time was spent. This helps us improve the test in the future.
- A short write-up (.txt is fine) containing answers to the following questions:
    - What assumptions or trade-offs did you make while implementing your solution?
    - If you were given another 4-6 hours, what changes or additions would you want to make?
    - How would you modify your program to handle a variable number of opponents? Custom decks?
    - What additions or clarifications to the instructions would have helped you implement your solution? Your feedback is appreciated!

Please organize your files into a single archive (.zip, .7z, tar, or gzip) for submission.

Have Fun!