**Poker Showdown User Guide**

**Problem Description**  
Poker is a common card game with many variations. We would like to build a library which will take poker hands as input and output the player with the winning hand for a particular showdown.

**Input**

The library accepts data input in the following format:

<Name>

<Hand>

There are two lines of data where <Name> is the name of the player (as a string of text) and <Hand> is a five-card poker hand in the following format:

* A comma separated list of cards
* Cards are represented with the card value first followed by the card suit
* Each card is up to three characters – one or two characters for the card value and a single character for the suit
* Card values are represented by the following values: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A
* Suites are listed as S, C, D, H for Spades, Clubs, Diamonds and Hearts respectively

**Example Input:**

Joe

AD, 5C, JH, 7S, 4C

*(interpreted as: Joe has a hand consisting of the following cards: Ace of Diamonds, Five of Clubs, Jack of Hearts, Seven of Spades and Four of Clubs)*

**Constraints:**

* Each player has only one hand of cards
* Each hand has only 5 cards
* Each card can only exist in one player’s hand
* Ace is considered the highest card value

**Poker Ranks used**

We only compare a subset of poker hand ranks within this library (see Terminology document for definitions):

* **Flush**
* **Three of a kind**
* **One pair**
* **High Card**

**Basic Algorithm:**

The basic algorithm for solving this problem is described as follows:

1. Read the input into a list of raw data objects
2. For each raw data item:
   1. Create a player using the name and cards to build the hand
   2. Evaluate the rank of the player’s hand
   3. Validate the player’s hand
   4. If the hand is valid add the player to the showdown
   5. Compare the player’s hand to the current winner
   6. If the player’s hand ranks higher they become the current winner
   7. If there is a tie, we add the player to the list of winners
3. Repeat until there are no more players to add to the showdown
4. Output the name of the current winner or winners if there’s a tie

**References:**

Poker is a well studied game with plenty of resources available for implementing Poker evaluation algorithms. I’d like to reference the following URL’s I used as part of my research and implementation for this project:

ADDA52 poker rules

<https://www.adda52.com/poker/poker-rules/cash-game-rules/tie-breaker-rules>

Nick Sayer’s blog - Algorithm for Evaluating Poker Hands

<http://nsayer.blogspot.com/2007/07/algorithm-for-evaluating-poker-hands.html>

Emory University Computer Science 170 course – Designing the Poker library: Checking for Poker hands

<http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/10/pokerCheck.html>

Stack Overflow – Simple histogram generation of integer data in C#

<https://stackoverflow.com/questions/926067/simple-histogram-generation-of-integer-data-in-c-sharp>

Stack Overflow – Algorithm that gives hand strength in poker

<https://stackoverflow.com/questions/43482791/algorithm-that-gives-hand-strength-in-poker>

Stack Overflow – The simplest algorithm for poker hand evaluation

<https://stackoverflow.com/questions/10363927/the-simplest-algorithm-for-poker-hand-evaluation>

Stack Overflow – Read two lines of data from a text file

<https://stackoverflow.com/questions/43883153/read-two-lines-of-data-from-a-text-file>

Code Buckets – Getting the Root Directory Path for .Net Core Applications

<http://codebuckets.com/2017/10/19/getting-the-root-directory-path-for-net-core-applications/>

Wikipedia – Glossary of Poker Terms

<https://en.wikipedia.org/wiki/Glossary_of_poker_terms>