**Kevin Jan C. Alvarez.**

**Assumptions**

**Input Parameters**

* Based on the provided document, the library will accept input of Player name and card at hands.
* Assuming that these are the input.
* I will design the code to accept list of players that has the following structure:
  + Player Name
  + Cards at hand

as input parameters.

**Output Parameters**

* Based on the provided documentthe library needs to return the following values:
  + Name of the winner(s)
  + Cards at hand of the winner(s)
  + Poker Hand of the winner(s)
* Assuming that these are the needed outputs.
* I design the library to return list of winners.

**Randomizer**

* Based on the provided document, the library will accept input of Player name and card at hands.
* Assuming that these are the input.
* I will create a method that will generate deck of cards and distribute on each player for unit test and can be used by the calling application to generate cards at hand automatically.

**Invalid Card Value**

* Assuming that the system will use this library will create a method or used the library’s existing automatic deck generator.
* Handling common error like input of invalid combination of value and/or suit will not be handled by the library see valid suit and value below.
  + Value = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11(JACK), 12(QUEEN), 13(KING) and 14 (ACE)
  + Suits = H (HEARTS), S (SPADES), D (DIAMONDS) and C (CLUBS)

**Value of Jack, Queen, King and Ace**

* Assuming that the system will use c# Random() method or and algorithm that will generate random integer value to generate the deck.
* The value of Jack, Queen, King and Ace will be numeric.
* The library doesn’t need to evaluate the following characters (J, Q, K and C) and transform to numeric value.

**Draw / Multiple Winners**

* Using the traditional High Poker Hand Ranks
* I will only evaluate the card value. Suit doesn’t matter
* If both player have the same ranking, with the same card value, there can be more than winners or all players can be winners (draw), some sample rankings are Royal Flush, Straight flush, straight etc.

**Straight**

* Using multiplication of prime numbers to calculate the ranking of the cards in general
* Assuming that the card ranking is straight
* Since the combinations are prime numbers, to get the same product I need to have the exact same combination. It makes the product and combination unique.
* Only evaluate the possible product of a straight combination below:
  + 2310 - 2 3 4 5 6 Combination
  + 15015 - 3 4 5 6 7 Combination
  + 85085 - 4 5 6 7 8 Combination
  + 323323 - 5 6 7 8 9 Combination
  + 1062347 - 6 7 8 9 10 Combination
  + 2800733 - 7 8 9 10 J Combination
  + 6678671 - 8 9 10 J Q Combination
  + 14535931 - 9 10 J Q K Combination
  + 31367009 -10 J Q K A Combination
* If exists on the combination above, it is a straight.

**Flush**

* Using multiplication of prime numbers to calculate the suit of the cards to identify if it is a flush
* Since the combinations are prime numbers, to get the same product I need to have the exact same combination. It makes the product and combination unique.
* Only evaluate the possible products of a flush below:
  + 147008443 – all hearts
  + 229345007 – all spades
  + 418195493- all diamonds
  + 714924299 – all clubs
* If exists on the combination above, it is a flush.

**Royal Flush**

* Using the High Poker Hand combination
* Evaluate if the card is flush and straight and has the highest possible product on a straight combination (*31367009* which means there an ACE on the combination)
* If all parameters are met, it is a royal flush.

**Hand Ranking Unit Test**

* Using the High Poker Ranking.
* Assuming that the rank is correct.
* I don’t need to evaluate each ranking to each other.
* I just need to evaluate the current rank to rank – 1 and so on.
  + Royal Flush > Straight Flush
  + Straight Flush > Four Of A Kind and so on ….

**Test Cases - AutomatedTest**

**H** – Hearts, **S** – Spades, **C** – Clubs, **D** – Diamonds

* Test method name: *CompareCardsAtHand\_TwoRoyalFlushOneStraight\_TwoWinners()*
* Given the following player hands, the winners are Player\_1 and Player\_2

*Player\_1: 10C, JC, QC, KC, AC (Royal Flush)*

*Player\_2: 10H, JH, QH, KH, AH (Royal Flush)*

*Player\_3: 2H, 3D, 4C, 5S, 6H (Straight)*

* Test method name: *CompareCardsAtHand\_TwoStraightFlushOneFourOfAKind\_StraightFlushWithHigherCardValueWin()*
* Given the following player hands, the winner is Player\_2

*Player\_1: 2C, 3C, 4C, 5C, 6C (Straight Flush)*

*Player\_2: 8H, 9H, 10H, JH, QH (Straight Flush with higher values)*

*Player\_3: 2H, 3D, 4C, 5S, 6H (Straight)*

* Test method name: *CompareCardsAtHand\_TwoStraightFlushOneFourOfAKind\_StraightFlushWithHigherCardValueWin()*
* Given the following player hands, the winner is Player\_2

*Player\_3: 2C, 2H, 2D, 2S, 6C (Four of a kind)*

*Player\_2: 8H, 8C, 8S, QH, QS (Full House)*

*Player\_1: 3C, 4C, KC, JC, 7C (Flush)*

* Test method name: *CompareCardsAtHand\_FullHouseFlushStraight\_FullHouseWin()*
* Given the following player hands, the winner is Player\_2

*Player\_1: 2C, 3C, 5C, 8C, 9C (Flush)*

*Player\_2: JH, JC, JS, QH, QS (Full House)*

*Player\_3: 2H, 3D, 4C, 5S, 6H (Straight)*

* Test method name: *CompareCardsAtHand\_FlushStraightThreeOfAKind\_FlushWin()*
* Given the following player hands, the winner is Player\_1

*Player\_1: 2C, 3C, 5C, 8C, 9C (Flush)*

*Player\_2: JH, JC, JS, QH, KS (Three of a kind)*

*Player\_3: 2H, 3D, 4C, 5S, 6H (Straight)*

* Test method name: *CompareCardsAtHand\_TwoStraightThreeOfAKind\_TwoStraightWinners()*
* Given the following player hands, the winners are Player\_1 and Player\_3

*Player\_1: 2C, 3H, 4S, 5D, 6C (Straight)*

*Player\_2: AH, AC, AS, KH, 7S (Three of a kind)*

*Player\_3: 2H, 3D, 4C, 5S, 6H (Straight)*

* Test method name: *CompareCardsAtHand\_ThreeOfAKindTwoPairOnePair\_ThreeOfAKindWin()*
* Given the following player hands, the winner is Player\_1

*Player\_1: 2C, 2H, 2S, 5D, 6C (Three of a kind)*

*Player\_2: QH, QC, KS, KH, 7S (Two Pair)*

*Player\_3: AH, AD, 4C, 5S, 6H (One Pair)*

* Test method name: *CompareCardsAtHand\_TwoTwoPairOnePair\_HigherTwoPairWin()*
* Given the following player hands, the winner is Player\_2

*Player\_1: 2C, 2H, 5S, 5D, 6C (Two Pair)*

*Player\_2: QH, QC, KS, KH, 7S (Higher Two Pair)*

*Player\_3: JH, JD, 4C, 5S, 6H (One Pair)*

* Test method name: *CompareCardsAtHand\_TwoOnePairAndHighCard\_HigherOnePairWin()*
* Given the following player hands, the winner is Player\_3

*Player\_1: 10C, 10H, 5S, 6D, 8C (Two Pair)*

*Player\_2: 2H, AC, 4S, 9H, QS (Higher Two Pair)*

*Player\_3: JH, JD, 4C, 5C, 6s (One Pair)*

* Test method name: *CompareCardsAtHand\_AllHighCards\_HighestHigCardWin()*
* Given the following player hands, the winner is Player\_1

*Player\_1: 2C, 4H, 5S, 8D, AC (Highest High Card - AC)*

*Player\_2: 3H, 5C, JS, QH, KS (High Card - KS)*

*Player\_3: 10H, 2D, 8C, QC, JH (High Card - QC)*

* Test method name: *CompareCardsAtHand\_RandomCardGenerator\_ThereWillBeWinner()*
* Given the randomness of the card that will be distributed in each player we can’t identify the result but the possibilities are:
  + 1 winner
  + 2 winners
  + Draw – All winner