CS 101
Fall 2015
Program 8
Algorithm Due : Dec. 6 2015
Program Due : Dec. 13 2015
Whist player class

Whist is a card game for 4 players. It is a forerunner of Bridge, but unlike Bridge, it has no bidding phase. It is played with a standard 52-card deck (see appendix at the bottom of the assignment if you're unfamiliar with the standard deck). There is no ranking among the suits. Play proceeds as follows:

- The game requires 4 players, who play in 2-person partnerships. Partners sit across the table from each other. Customarily, the players are identified by compass directions (North, South, East, West). North and South are partners, as are East and West.
- A dealer is chosen by lot. The dealer deals the entire deck, one card at a time, starting with the player to the dealer's left and proceeding clockwise, until the deck is exhausted and each player has a hand of 13 cards.
- The last card dealt to the dealer is turned face up. The suit of this card is *trump* for this hand. (After the first trick, the dealer puts this card back in her hand.)
- Each hand consists of 13 *tricks*, in which each player plays one card. The first trick is led by the player to the dealer's left and play proceeds to the left.
- Each player plays one card. Players must follow suit (play the same suit that was led) if they have any cards of that suit. If they have no cards of that suit, they may play a card of the trump suit, or may discard (play any other card in their hand).
- When each player has played, the highest card played of the trump suit takes the trick. If no trumps were played (which is more often the case), the highest card played *of the suit led* takes the trick. A card that is not the suit led or trump cannot take the trick.
    - Suppose Spades are trump, and East leads the 6 of Clubs. Other players play the Ace of Diamonds, Ace of Hearts, and King of Diamonds. Since no trump were played, and the other players were apparently unable to follow suit, the 6 of Clubs takes the trick. On the other hand, if the 2 of Spades were played instead of the King of Diamonds, it would take the trick, as it was trump.
- The winner of one trick leads the next. The player leading the trick may lead with any card they choose.
- When all cards have been played, each partnership counts the tricks they have taken. The winning partnership scores 1 point for each trick *over 6* that they took. (If they took 7 tricks, they score 1 point; 8 tricks scores 2 points; and so on).
- The game is 5 points. As only one partnership can score each hand, ties are impossible.
- If neither partnership has yet won the game, the deal passes to the left and another hand is played.

You will write a Player object (class). This class will have a method called PlayCard(). The parameter list for this method is rather long:

- A tuple of cards. These are the cards in your hand.

- A (possibly empty) list of cards that have been played so far this trick. (If the list is empty, you need to lead for this trick.) Cards are listed in the order played.
- A string (one of 'S', 'H', 'D', 'C') specifying trump for this hand.
- A string (one of 'N', 'S', 'E', 'W') specifying which position you are playing.
- A (possibly empty) list of tricks that have been played so far this hand. Each is a tuple consisting of: 4 cards (the cards that were played), a string "N", "S", "E", "W" specifying who led that trick, and another string (same format) specifying who won that trick.
- The current score of the game; NS score, then EW score.
- The PlayCard() method should not have defaults for any of these parameters.

Your PlayCard method must select and return a card from your hand to play. If it is not leading, it must follow suit if possible. It must return a Card, not a string, integer, or other representation of the card.

That's good for a B.

For an A, you must implement some kind of strategy in your class. Document this strategy in your code. How do you select which card to play? There are several possibilities.

- You may choose to always play the highest card available.
- You may decide to always play trump if you can (to maximize the number of tricks taken).
- On the other hand, if a card's already been played that's higher than anything you have in that suit, then you can't take the trick (you have to follow suit if you can), and should probably play your lowest card in that suit.
- On the other hand, if your partner led an Ace in a suit you don't have, that will probably take the trick and you should save your trump for when it would be more needed.
- BUT, if it's late in the hand and there are only a few cards of that suit left, an opponent may play a trump.
- One strategy is to "draw trumps" by leading with your high trumps to force opponents to play their trumps early, so they can't use them to take tricks later in the game. This assumes that you have the high trumps. If you only have low trumps, or only have a few, try to look for better opportunities.
- If you only have a few cards of a suit, you may decide to discard them as soon as possible, so you can play trump the next time that suit is led.
- With careful tracking of what cards have been played, it's possible to work out how many cards of a particular suit are unplayed, and where some of the cards may lie. (If the last time an opponent led trump, your partner discarded, then your partner is out of trumps. If 12 trumps have already been played and you have 1 trump in your hand, then you have the last trump.)

You must use some method beyond playing the first (or last) legal card found or randomly choosing from legal cards. You can base it on the rank of the cards in your hand, on what cards have been played, etc. (The actual game has strict rules against going through previous tricks to see what's been played, but your object will have that information available, since it won't be 'watching' as the hand is played. You don't have to use that information for your strategy, but it's available if you want to use it.)

Support code:

- Your program will use the Card and DeckOfCard classes attached. DO NOT MODIFY THESE CLASSES. The classes used will have the same interface—the same public methods will be available and will have the same parameters. **Do not assume the private/internal data elements of the class will be the same.** Read the docstrings of the classes carefully; these are not quite the same as versions developed in class. Put the Card.py file in the same folder as your code. Your program will `import Card` early in the program.

- Your file should be named Player.py. Your class name should be Player.
- Your Player should have an __init__(self) method. You may add whatever other methods or data you please, but must have a PlayCard() method with the interface specified above.
- A test script and file will be posted. The script will pass various information to the object and verify that it returns a Card from the hand, and the Card returned is a legal play. All submissions that pass the testing without errors will be entered into a tournament. Details will be announced as the deadline approaches (and the instructors get the various test scripts written).

APPENDIX: The standard deck of cards (US/UK)
The standard deck consists of 52 cards, comprising 4 suits of 13 cards each. The suits are: Spades, Hearts, Diamonds, Clubs. The suits are printed in contrasting colors: Spades and Clubs are Black suits, Hearts and Diamonds are Red suits. Color is not significant in Whist. There is no ranking among the suits in Whist.

The ranks of the cards are: Ace (high), King, Queen, Jack (often called the Knave in UK), 10, 9, 8, 7, 6, 5, 4, 3, 2 (low). In Whist the Ace is always high; this varies in some games.

Some games also have one or two *Jokers*, which are outside the normal ranking. They have no rank and no suit (or constitute their own). Whist does not use Jokers.