## Money-Media Programming Assignment

Here are descriptions of three classes in an object-oriented program that you will write.

=== Deck ===

- The Deck class represents a standard 52-card deck; Ace high
- Each card is in one of two states -- dealt (D) and not-yet-dealt (ND)
- Cards are ordered. The D and ND cards each have their own order
- When created, all 52 cards are ND, and are in order, by suit
  (clubs 2 through A, diamonds 2 through A,
  hearts 2 through A, spades 2 through A)

methods:
  dealOne() -- moves the top card from ND to D and returns the card
  print() -- prints non-dealt cards and dealt-cards (in order, as separate lists)
  shuffle() -- Shuffles the deck randomly

Make your own shuffle algorithm. Be prepared to explain how it works.

=== Card ===

- The Card class represents a single card

methods:
  print() -- prints the type of card

=== Hand ===

- The Hand class represents a set of cards. From 0 to 52 cards, total
- Cards are always in a definite order

methods:
  print() -- prints the hand (in order)
  addCard(card) -- adds a card to the hand

  sortBySuit() -- sorts the cards by suit (see intro for suit order), and then by value (see intro for value order)
  sortByValue() -- sorts by value, then by suit

Write your own custom sort algorithm.  Be prepared to explain how it works.

hasStraight(len, sameSuit) -- returns true if hand
contains a straight of the given length.
If sameSuit is true, counts only straights
with cards in the same suit (flushes);
If sameSuit if not true, any straight is counted

==========

1) Write a program that

- Implements the classes above and creates and shuffles a deck and then deals two or more 5-card hands from the deck.

Guidelines:
==========

- Write all the code yourself including the code to shuffle and and sort

- Should be a simple command-line utility, not a gui app. If you generate any graphs, you can use any utility, such as a spreadsheet, matlab, matplotlib, gnuplot, etc.

- Code must be written in php

- Design is important; get the interfaces right

- Code readability is important

- A simple implementation that gets the job done (and meets other requirements) is better than a complex implementation.

- Comments not necessary unless there is something that might not be obvious to a reader. Make a note of any assumptions that may not be obvious from a method signature.

- Careful error checking not required, but if error checking helps you develop the code, it's fine to leave it in.

- Don't over-engineer -- this is a limited problem, not the start of a

commercial poker system

- If you write test code to make sure things are working, go ahead and include that.