Santa Clara University
School of Engineering

CSEN 79L: OOP and Adv. Data Structure

Lab 2: Playing Card, Deck, and Poker Ranking

This lab exercises multi-file development and implementation of simple classes with some operator overloading. It should also get you proficient with development tool chain: make, debugger, etc.

It also gets you familiar with the game of Poker, particularly its ranking system and the basic statistics.

In the provided partial implementation, there is a "card" class implemented in "card.h" and "card.cxx". This class abstracted the simple concept of one playing card that has a "suit" and "rank." (It did not abstract the concept of "Joker.") In this implementation, Ace is ranked numerically 14, King 13, Queen 12, Jack 11, and other cards their face value until deuce as numerically 2. The class overloaded formatted output "<<" operator to display a card. The file "cardmain.cxx" tests the class. This class implemented the "cast operator" int() so that a "card" can be converted to an integer for ease of comparing and sorting. You should study this class before you proceed.

Part 1: Deck and Shuffling

1. Implement "deal" and "shuffle" functions for the provided "Deck" class that abstract the concept of a standard 52-card deck of playing cards (no jokers). Both functions are defined in shuffle.cxx. Use Fisher-Yates algorithm for shuffling.
2. Note the Deck constructors set the "guard" value. The "deal" function should deal out the top of the deck until there's less than "guard" cards left. At that point, it reshuffles then continue dealing.
3. Write your test code in deckmain.cxx.
4. The provided Makefile will build both card and deck test programs.

Part 2: 5-Card Poker

Single deck Poker hands are categorized into the following "ranks" in descending order (higher one "wins" lower): Straight Flush, Four of a Kind, Full House, Flush, Straight, Three of a Kind, Two Pairs, Pair, and High Card. Understand their definitions from the Internet. If you wish, read the companion "Technique for Ranking Poker Hands" to get started on this.

5. Complete the member function "rankHand" that categorizes the dealt hand into one of the ranking categories. This function uses helpers to test if a hand is in the ranked category. Replace the stud implementation for 5 "isXXX" functions from line 59 to 63.
6. Finish functions PokerHands and PokerStats in pokermain.cxx.
7. PokerHands prints one "sample" hand for each rank category. For example:

```
♠Q♣5♦5♣2♠2 Two Pairs
♥K♣Q♥9♥5♦2 High Card
♦8♠7♣6♥6♥2 Pair
♣9♣7♦7♥7♥3 Triple
♥10♠9♥8♦7♦6 Straight
♦A♣J♦J♥J♠J Four of a Kind
♣K♣Q♣10♣7♣3 Flush
```

```
♣A♥A♦3♥3♠3 Full House
♥Q♥J♥10♥9♥8 Straight Flush
```

(The simple way to do this is by setting a "flag" for each of the ranked category to track if this rank has been seen.)

8. PokerStats computes statistics for each ranking. Do it in Monte Carlo style by keep on dealing random hands and record the occurrences for each rank. Stop periodically to see if the statistics have "stabilized." (That the percentages for each rank appear not changing much anymore.) Print out the percentages of each rank, also the average time consumed doing so for a nominal number of hands. For example:

```
Dealt 8795000 hands. Elapsed Time: 3.60186 seconds.
Average 0.0204767 seconds per 50k hands
 Straight Flush:      104  0.00%
 Four of a Kind:     2490  0.03%
     Full House:    13449  0.15%
          Flush:    17938  0.20%
       Straight:    27404  0.31%
         Triple:   194476  2.21%
      Two Pairs:   426311  4.85%
           Pair:  3733156 42.45%
      High Card:  4379666 49.80%
```