NAME: **login id:**

Precept:

COS 126 Midterm 2 Programming Exam Fall 2014

This part of your exam is like a mini-programming assignment. You will create four programs, compile them, and run them on your laptop, debugging as needed. This exam is open book, open browser—but only our course website and booksite! Of course, no internal or external communication is permitted (e.g., talking, email, IM, texting, cell phones) during the exam. You may use code from your assignments or code found on the COS126 website. When you are done, submit your program via the course website using the submit link for Programming Exam 2 on the Assignments page.

Grading. Your program will be graded on correctness, clarity (including comments), design, and efficiency. You will lose a substantial number of points if your program does not compile or if it crashes on typical inputs.

Even though you will electronically submit your code, you must turn in this paper so that we have a record that you took the exam and signed the Honor Code. Print your name, login ID, and precept number on this page (now), and *write out and sign* the Honor Code pledge before turning in this paper. Note: It is a violation of the Honor Code to discuss this midterm exam question with anyone until after everyone in the class has taken the exam. You have 90 minutes to complete the test.

"I pledge my honor that I have not violated the Honor Code during this examination."

_____

signature _____

Part 1	/ 7
Part 2	/16
Part 3	/ 6
Part 4	/ 1
TOTAL	/30

Part 1 (7 points). Begin by downloading the files `Deck.java` and `FiveHands.txt` at

http://www.cs.princeton.edu/~cos126/docs/data/Poker

Your task now is to modify `Deck.java` to create a new program `RandomPokerHands.java` that (*i*) accepts an integer `N` from the command line, (*ii*) initializes a deck of cards, and (*iii*) performs the following operation `N` times: shuffle the deck and "deal" a five-card poker hand (by printing the first five cards in the deck). Print each hand on a separate line, using a two-character code for each card with `C`, `D`, `H`, `S` for `Clubs`, `Diamonds`, `Hearts`, and `Spades` (respectively) and `T`, `J`, `Q`, `K`, `A` for `Ten`, `Jack`, `Queen`, `King`, `Ace` (respectively). Note that you need to make sure that the suit and rank entries are one character each. For example, your program should produce output like the following:

```
% java-introcs RandomPokerHands 10
7H 3S TS QH 9D
2C 2S 7C 4C AS
JS 3S KD 7D 8H
5S 2C 2S 8H TS
QD JH KC 5C 4H
JC 7D 7S KC 8S
JC 3H AS 4C AD
3S TH KS 8H AH
AH 5C AC 5H 8C
5S 8S 7S TH 3C
```

Your output will differ from this because of different initial conditions for the random number generator. We will use our solution to this to create test files to evaluate your solutions to Parts 2, 3, and 4. You may find it convenient to use it yourself to help debug your programs.

In particular, you will use `FiveHands.txt` in Parts 2, 3, and 4.

```
% more FiveHands.txt
7H 8D 2C 7S 8H
AC 5H AD 5C AS
JD 6D 7D AD 5D
2S 2H QC JC TD
9H 2S 2C JH QH
```

Submission. Submit your file `RandomPokerHands.java` via Dropbox at

https://dropbox.cs.princeton.edu/COS126_F2014/Exam2

(This is the submit link for Programming Exam 2 on the Assignments page.) Be sure to click the Check All Submitted Files button to verify your submission.

Grading. Your program will be graded on correctness and clarity (including comments).

Part 2 (16 points). A *flush* is a poker hand where all cards are the same suit. A *full house* is a poker hand with three cards of one value and two cards of another value. Your task is to write a Java class that implements the following API:

`public class`	`PokerHand`	
	`PokerHand()`	read a 5-card poker hand from `StdIn`
`boolean`	`flush()`	is this hand a flush?
`boolean`	`fullHouse()`	is this hand a full house?
`String`	`toString()`	string representation of this hand (sorted)

Include a test client that reads poker hands from `StdIn` and prints out and identifies the flushes and full houses. Your program must behave as follows:

```
% java-introcs PokerHand < FiveHands.txt
Full house: 5C 5H AC AD AS
     Flush: 5D 6D 7D AD JD
```

Note: You can sort your hand, as required, in the constructor by using the static method `java.util.Arrays.sort()`. If you keep the hand as an instance variable in an array named `hand`, just call `java.util.Arrays.sort(hand)`. Sorting your hand can simplify your code for `fullHouse()`. Do not worry that the tens and the face cards do not appear in proper order in this sort.

You may wish to use your program from Part 1 to help in debugging, as follows:

```
% java-introcs RandomPokerHands 2000 | java-introcs PokerHand
     Flush: 4H 7H JH KH QH
Full house: 7C 7S JC JD JH
     Flush: 2C 7C 8C 9C TC
          . . .
```

Note that this test checks that your program properly identifies the hands that it prints, but would *not* identify a bug where you miss a flush or a full house in the input.

Submission. Submit your file `PokerHand.java` via Dropbox at

`https://dropbox.cs.princeton.edu/COS126_F2014/Exam2`

(This is the submit link for Programming Exam 2 on the Assignments page.) Be sure to click the Check All Submitted Files button to verify your submission.

Grading. Your program will be graded on correctness and clarity (including comments). The constructor, methods, and test client will be 2 to 4 points each.

Part 3 (6 points). Download `Queue.java` from the booksite and implement a `Queue` and `PokerHand` client `PokerHandCount.java` that consists of a single `main()` static method that maintains two queues, one with flushes and one with full houses. Your program should read hands from standard input, saving the flushes and full houses on their respective queues. After the input is exhausted, it should print the number of flushes followed by the first 5 flushes found (one per line), then the number of full houses followed by the first 5 full houses found. If there are fewer than 5 hands in a queue, print out all the hands in the queue. Feel free to use the constant "5" everywhere, (rather than bothering with a `final` variable name.) Your program should produce output like the following:

```
% java-introcs RandomPokerHands 2000 | java-introcs PokerHandCount
    Flushes: 10
          2H 4H 5H 7H JH
          3H 6H 8H JH QH
          2C 3C 5C 8C QC
          4S 5S 8S 9S KS
          5C 7C 9C AC TC
Full houses: 4
          7C 7H 8C 8D 8H
          5C 5D 7C 7D 7H
          2C 2H 2S 9H 9S
          9H 9S TC TD TH
```

Note: When you run your program for, say, 1 million hands (as we will), you will get the idea that a flush is more likely than a full house, which is why a full house beats a flush in poker.

Submission. Submit your file `PokerHandCount.java` (we have `Queue.java`, so you do not need to submit that) via Dropbox at

`https://dropbox.cs.princeton.edu/COS126_F2014/Exam2`

(This is the submit link for Programming Exam 2 on the Assignments page.) Be sure to click the Check All Submitted Files button to verify your submission.

Grading. Your program will be graded on correctness and clarity (including comments). It will also be graded on using the `Queue` correctly and efficiently, and for having no unnecessary code.

Part 4 (1 point). Be sure that you have successfully completed Parts 1, 2 and 3 before attempting this part of the exam.

Implement a Java class `PokerHandWild` that has the same API as Part 2, except that it supports a game with "deuces wild" where any 2 can be made to match any card. With wild cards, the first hand in `FiveHands.txt` is a full house and the fourth is a flush. Your program must behave as follows:

```
% java-introcs PokerHandWild < FiveHands.txt
Full house: 2C 7H 7S 8D 8H
Full house: 5C 5H AC AD AS
     Flush: 5D 6D 7D AD JD
     Flush: 2C 2S 9H JH QH
```

Note: If you were to run this program with a client like the one in Part 3 for, say, 1 million hands (as we will), you might learn something: wild cards dramatically change the rankings of the probability of occurrence of poker hands. No wonder people get confused about whether or not a full house beats a flush!

Submission. Submit your file `PokerHandWild.java` via Dropbox at

`https://dropbox.cs.princeton.edu/COS126_F2014/Exam2`

(This is the submit link for Programming Exam 2 on the Assignments page.) Be sure to click the Check All Submitted Files button to verify your submission.

Grading. Your program will be graded on correctness and clarity (including comments).