CSE 142 SPRING 2011 FINAL EXAM SOLUTIONS

1. Expressions (6 points)

For each expression in the left-hand column, indicate its value in the right-hand column.
Be sure to list a constant of appropriate type (e.g., `7.0` rather than `7` for a `double`, `String`s in "quotes"). If the expression is illegal, then write "error".

Expression	Value
`1 / 2 * 4 + 2 / 1`	`2`
`(3 < 5 < 10) && (3 > 1)`	`error`
`2.8 + 1.2 - 3 % 6 + 2 * 3 % 3`	`1.0`

2 points each for the first two, all or nothing
2 points for the last one (-1 if wrong type)

2. Array Mystery (10 points)

Consider the following method:

```
public static void mystery(int[] array) {
    for (int i = array.length - 2; i > 0; i--) {
        array[i] = array[i] + array[i-1] + array[i + 1];
    }
}
```

Indicate in the right-hand column what values would be stored in the array after the method `mystery` executes if the integer array in the left-hand column is passed as a parameter to `mystery`.

Array	Final Contents of Array
`int[] a1 = {8, 9};` `mystery(a1);`	`{8, 9}`
`int[] a2 = {7, 1, 1};` `mystery(a2);`	`{7, 9, 1}`
`int[] a3 = {5, 4, 3, 2, 1};` `mystery(a3);`	`{5, 22, 13, 6, 1}`
`int[] a4 = {1, 2, 3, 4, 5};` `mystery(a4);`	`{1, 20, 17, 12, 5}`

1 points per array for first two, all or nothing
4 points per array for second two, -2 per wrong/missing # (max -4 per array)

3. Reference Semantics Mystery (10 points)
What does the following program print?

```
import java.util.*;

public class ReferenceMystery {
   public static void main(String[] args) {
      int[] a = { 1, 2, 3, 4, 5 };
      int x = 0;

     System.out.println(x + " " + Arrays.toString(a));

      x++;
      a[x] = 12;
      mystery(a, x);
      System.out.println(x + " " + Arrays.toString(a));

      x++;
      a[x] = 34;
      mystery(a, x);
      System.out.println(x + " " + Arrays.toString(a));
   }

   public static void mystery(int[] a, int x) {
      x++;
      a[x] = 56;
      System.out.println(x + " " + Arrays.toString(a));
   }
}
```

```
0 [1, 2, 3, 4, 5]
2 [1, 12, 56, 4, 5]
1 [1, 12, 56, 4, 5]
3 [1, 12, 34, 56, 5]
2 [1, 12, 34, 56, 5]
```

2 points per line (1 point for left number, 1 point for array)
-1 (instead of -2) if lines 2 and 3 are wrong but the same (same goes with lines 4 and 5)

4. Mystery (3 points)

What does the following program print?

```
public class Mystery {
    public static void main(String[] args) {
        int a = 3;
        int b = 7;
        mystery(a, b);
        System.out.println(a + " " + b);
    }

    public static void mystery(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
    }
}
```

```
3 7
```

3 points, all or nothing

5. Inheritance Mystery (10 points)

Assume that the following classes have been defined:

```
public class Butters extends South {
    public void methodB() {
        System.out.print("Butters B ");
    }

    public String toString() {
       return "Loo loo loo";
    }
}

public class Wendy extends South {
    public void methodB() {
        System.out.print("Wendy B ");
    }

    public String toString() {
        return "Wendy";
    }
}
```

```
public class South {
    public void methodA() {
        System.out.print("South A ");
        methodB();
    }

    public void methodB() {
        System.out.print("South B ");
    }

    public String toString() {
        return "South Park";
    }
}

public class Stan extends Wendy {
    public void methodB() {
        System.out.print("Stan B ");
        super.methodB();
    }
}
```

Given the classes above, what output is produced by the following code?

```
South[] park = { new Wendy(), new Butters(), new South(), new Stan() };
for (int i = 0; i < park.length; i++) {
    park[i].methodA();
    System.out.println();
    park[i].methodB();
    System.out.println();
    System.out.println(park[i]);
    System.out.println();
}
```

```
South A Wendy B
Wendy B
Wendy

South A Butters B
Butters B
Loo loo loo

South A South B
South B
South Park

South A Stan B Wendy B
Stan B Wendy B
Wendy
```

1 point for 4 separate sections
2 points per section for first three (-1 per wrong word, max -2)
3 points for last section (-1 per wrong word, max -3)
-2 for quotes, consistent extra newlines

6. Programming (16 points)

Write a static method `blackjack` that prints random numbers in the range of 1 – 10 until the sum of all integers generated are greater than or equal to 17. The sum is subsequently printed. If the sum is greater than 21, then the method prints out `"Busted!"`; if the sum is equal to 21, then the method prints out `"BLACKJACK!"`. The method returns whether the sum was less than or equal to 21.

Call	Prints	Returns
`blackjack()`	`2 4 6 10 = 22 Busted!`	`false`
`blackjack()`	`2 2 1 5 7 = 17`	`true`
`blackjack()`	`9 5 7 = 21 BLACKJACK!`	`true`
`blackjack()`	`5 6 3 2 10 = 26 Busted!`	`false`
`blackjack()`	`10 5 4 = 19`	`true`

As this method has an element of randomness to it, you are to copy the format of the output, not the exact output of the sample calls.

```
public static boolean blackjack() {
   Random rand = new Random();

   int sum = 0;
   while (sum < 17) {
      int card = rand.nextInt(10) + 1;
      System.out.print(card + " ");
      sum += card;
   }

   System.out.print("= " + sum);
   if (sum > 21) {
      System.out.print(" Busted!");
      return false;
   } else if (sum == 21) {
      System.out.print(" BLACKJACK!");
   }

   return true;
}
```

header
- 1 return value
- 1 name

random
- 1 creates Random object
- 1 calls nextInt
- 1 calls nextInt with proper range

cards
- retrieves cards until >= 17
  - 1 attempt
  - 1 correct
- 2 proper cumulative sum

printout
- 2 conditionally prints BLACKJACK! and Busted!
- 1 prints # and spaces

return
- 1 has a return
- 1 returns true and false somewhere

2 works (can still get this point if minor error above)

7. Programming (10 points)

Assume that the following method exists:

```
// returns the order number of each letter (case-insensitive)
public static int charToIndex(char letter)
```

For example:

Call	Returns
`charToIndex('a')`	1
`charToIndex('A')`	1
`charToIndex('e')`	5
`charToIndex('Z')`	26

Write a static method `isAnagram` that accepts two words as parameters and returns whether the words are anagrams of each other. An anagram of a word is another word that uses the exact same letters. For example, "asleep" and "please" are anagrams as are "dad" and "add".

Call	Returns
`isAnagram("asleep", "please")`	`true`
`isAnagram("dad", "add")`	`true`
`isAnagram("yes", "no")`	`false`
`isAnagram("yes", "yea")`	`false`

Hint: Tally letters.

```
public static boolean isAnagram(String word1, String word2) {
   int[] letterCount1 = new int[26];
   int[] letterCount2 = new int[26];

   if (word1.length() != word2.length()) {
      return false;
   }

   for (int i = 0; i < word1.length(); i++) {
      char letter1 = word1.charAt(i);
      char letter2 = word2.charAt(i);
      letterCount1[charToIndex(letter1) - 1]++;
      letterCount2[charToIndex(letter2) - 1]++;
   }

   return Arrays.equals(letterCount1, letterCount2);
}
```

header
- 1 return value
- 1 name/params

arrays
- 1 creates arrays of the right range (>= 26)
- 1 properly indexes array
- 1 compares arrays

loop
- 1 correct bounds

charToIndex
- 1 calls charToIndex
- 1 uses return value

2 works (can still get this point if minor error above)

header (if used nested for-loops)
- 1 return value
- 1 name/params

counting
- 1 counts
- compare counts
  - 1 attempt
  - 1 correct

loop
- 1 correct bounds

compares letters
- 1 attempt
- 1 correct (case-insensitive)

(lose 2 works points)

8. Programming (18 points)

(a) (11 points) Write a method named `printRating` that takes a string containing movie information as a parameter and prints out a summary of that information.

The string parameter will be formatted as follows:

<ratings> `X` *<title>*

For example, all of the following are proper examples of possible strings that could be passed to `printRating`:

```
7.0 7.3 8.0 8.0 8.2 7.2 X The Hangover Part II
8.2 8.5 8.9 8.9 X X-Men: First Class
7.5 8.3 7.4 6.9 8.7 7.9 X Bridesmaids
7.4 X Thor
```

Assume that every movie has at least one rating.

The summarized information will be formatted as follows:

<title>`:` *<average rating>* `(`*<# of ratings>* `ratings)`

The method also returns the number of ratings in the string.

Examples:

Method call: `printRating("7.0 7.3 8.0 8.0 8.2 7.2 X The Hangover Part II");`
Output: `The Hangover Part II: 7.616666666666667 (6 ratings)`
Returns: `6`

Method call: `printRating("8.2 8.5 8.9 8.9 X X-Men: First Class");`
Output: `X-Men: First Class: 8.625 (4 ratings)`
Returns: `4`

Method call: `printRating("7.5 8.3 7.4 6.9 8.7 7.9 X Bridesmaids");`
Output: `Bridesmaids: 7.783333333333332 (6 ratings)`
Returns: `6`

Method call: `printRating("7.4 X Thor");`
Output: `Thor: 7.4 (1 ratings)`
Returns: `1`

(NOTE: When you retrieve the movie title as a line, the space between the `X` and the title is retained. Do not worry about the extra space.)

(Write your solution on the next page.)

```
    public static int printRating(String line) {
       Scanner lineScan = new Scanner(line);

       int numRatings = 0;
       double totalRatings = 0;
       while (lineScan.hasNextDouble()) {
          numRatings++;
          totalRatings += lineScan.nextDouble();
       }

       lineScan.next();  // throw away token

       String title = lineScan.nextLine();
       System.out.println(title + ": " + (totalRatings/numRatings) + " (" +
numRatings + " ratings)");
       return numRatings;
    }
```

header
- 1 return value
- 1 name/params

Scanner
- 1 correct creation

text processing
- 1 properly disposes of extra token (.next() or .substring())
- 1 properly stops at "X" (hasNextDouble())

average rating
- 1 correct computation

printout
- 1 correct

return
- 1 correct

3 works (can still get this point if minor error above)

-1 int vs. double
-1 eliminates spaces in title

(b) (7 points) Using `printRating`, write a method named `printMovieFile` that takes a Scanner for an input file as a parameter. Each line of the input file contains movie information as specified in part (a) of this problem. The method should output the summarized information and end with the total number of ratings.

For example, if the input file `movies.txt` contained the following:

```
7.0 7.3 8.0 8.0 8.2 7.2 X The Hangover Part II
8.2 8.5 8.9 8.9 X X-Men: First Class
7.5 8.3 7.4 6.9 8.7 7.9 X Bridesmaids
7.4 X Thor
```

Then the call `printMovieFile(new Scanner(new File("movies.txt")))` would output

```
 The Hangover Part II: 7.616666666666667 (6 ratings)
 X-Men: First Class: 8.625 (4 ratings)
 Bridesmaids: 7.783333333333332 (6 ratings)
 Thor: 7.4 (1 ratings)

Total ratings: 17
```

```
public static void printMovieFile(Scanner data) {
   int numRatings = 0;
   while (data.hasNextLine()) {
      numRatings += printRating(data.nextLine());
   }
   System.out.println();
   System.out.println("Total ratings: " + numRatings);
}
```

header
    1 all or nothing
Scanner
    1 uses Scanner properly
printRating
    1 uses printRating
    1 adds return value to cumulative sum
printout
    1 correct
2 works (can still get this point if minor error above)

9. Programming (16 points)

Write a class called `Koala` that extends the `Critter` class according to the following specifications:

Constructor	`public Koala(boolean fighter)`
Color	`Color.GRAY`
toString	default value
getMove	Always infect if an enemy is in front and this koala is a fighter (when `fighter` is true) otherwise hop three times if possible otherwise pick a random direction (left or right) to turn

```
public class Koala extends Critter {
   private int hops;
   private boolean fighter;

   public Koala(boolean fighter) {
      hops = 0;
      this.fighter = fighter;
   }

   public Color getColor() {
      return Color.GRAY;
  }

   public Action getMove(CritterInfo info) {
      if (info.getFront() == Neighbor.OTHER && fighter) {
         hops = 0;
         return Action.INFECT;
      } else if (info.getFront() == Neighbor.EMPTY && hops < 3) {
         hops++;
         return Action.HOP;
      } else {
         hops = 0;
         Random rand = new Random();
         if (rand.nextInt(2) == 0) {
            return Action.LEFT;
         } else {
           return Action.RIGHT;
         }
      }
   }
}
```

class
- 1 correct class declaration

fields
- 1 int field
- 1 boolean field
- 1 proper initialization of fighter parameter

getColor
- 1 correct

getMove
- 1 proper check for other species
- 1 infects if fighter
- 1 proper check for empty and hop count
- 1 creates Random object
- 2 "coin flip"
- 2 returns something
- 1 resets hops correctly
- 2 works (can still get this point if minor error above)
  - (loses if adds and messes up toString)