CSc 15 Final Exam study Guide

My Exam Schedule > Fall 2015 > Sacramento State				
Class	Class Title	Exam Date	Exam Time	Exam Room
CSC 15-08 (85335)	Program Concept+Method I (Discussion)	12/15/2015, Tuesday	12:45PM - 2:45PM	Douglass Hall 108
CSC 15-04 (80169)	Program Concept+Method I (Discussion)	12/16/2015, Wednesday	8:00AM - 10:00AM	Folsom 1050
CSC 15-01 (80074)	Program Concept+Method I (Discussion)	12/17/2015, Thursday	3:00PM - 5:00PM	Acad Res Ctr 3004

Your final exam will cover chapters 1-8, in-class exercises, HW and lectures.

Your exam may contain multiple choice questions.

Your exam will contain coding, generating the output,..

Here are some samples problems, the difficulty level of your exam will be the same as the following questions

1. Write a program that accepts a number from the user and returns a string representing the factorial of that number. For example factorial(10) should return the string "9*8*7*6*5*4*3*2*1". factorial(-5) should return an empty string.

2. Write program that accepts a word containing the @ character as an input. If the word does not contain an @ character, then your program should keep prompting the user for the word. When the user enters the right word your program should output it on the screen.

3. Write a program that reads a sentence and output a message based on the last character in the sentence.

Last character	**output**
"."	declarative
"!"	exclamatory
"?"	interrogative

4. Write a program that accepts a temperature and outputs a probable season. Use the following table

Temperature	season
Greater than or equal 90	summer
Between 70 and 90	Spring

Between 50 and 70	fall
Below 50	winter
Above 110 or below -5	invalid

5. Write a method that accepts an array of double values and returns the percentage of the values above 90.

6. Write a program that reads 7 different values from the keyboard that represents the Dow Jones Average for 7 days. Your program should output the lowest value for those seven days and the number of the days on which the lowest value occurred.

7. write a method that accepts two arrays of String and return true if the two arrays are equal and return false otherwise.

8. Write a method called `allLess` that accepts two arrays of integers and returns `true` if each element in the first array is less than the element at the same index in the second array. Your method should return `false` if the arrays are not the same length. For example, if the two arrays passed are {45, 20, 300} and {50, 41, 600}, your method should return `true`. If the arrays are not the same length, you should always return `false`.

9. Write a method named `swapPairs` that accepts an array of strings as a parameter and switches the order of values in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on. For example, if the array initially stores these values:

```
String[] a = {"four", "score", "and", "seven", "years", "ago"};
swapPairs(a);
```

Your method should switch the first pair ("four", "score"), the second pair ("and", "seven") and the third pair ("years", "ago"), to yield this array:

```
{"score", "four", "seven", "and", "ago", "years"}
```

10. Write a method `isPalindrome` that accepts an array of `Strings` as its argument and returns `true` if that array is a palindrome (if it reads the same forwards as backwards) and `/false` if not. For example, the array `{"alpha", "beta", "gamma", "delta", "gamma", "beta", "alpha"}` is a palindrome, so passing that array to your method would return `true`. Arrays with zero or one element are considered to be palindromes.

11. Write a method called `mode` that returns the most frequently occurring element of an array of integers. Assume that the array has at least one element and that every element in the array has a value between `0` and `100` inclusive. Break ties by choosing the lower value.

For example, if the array passed contains the values {27, 15, 15, 11, 27}, your method should return 15. (*Hint:* You may wish to look at the `Tally` program from earlier in this chapter to get an idea of how to solve this problem.)

12. Write a method named `minGap` that accepts an integer array as a parameter and returns the minimum 'gap' between adjacent values in the array. The gap between two adjacent values in a array is defined as the second value minus the first value. For example, suppose a variable called `array` is an array of integers that stores the following sequence of values.

```
int[] array = {1, 3, 6, 7, 12};
```

The first gap is 2 (3 - 1), the second gap is 3 (6 - 3), the third gap is 1 (7 - 6) and the fourth gap is 5 (12 - 7). Thus, the call of`minGap(array)` should return 1 because that is the smallest gap in the array. Notice that the minimum gap could be a negative number. For example, if array stores the following sequence of values:

```
{3, 5, 11, 4, 8}
```

The gaps would be computed as 2 (5 - 3), 6 (11 - 5), -7 (4 - 11), and 4 (8 - 4). Of these values, -7 is the smallest, so it would be returned.

13. Write a method called `collapse` that accepts an array of integers as a parameter and returns a new array containing the result of replacing each pair of integers with the sum of that pair. For example, if an array called list stores the values {7, 2, 8, 9, 4, 13, 7, 1, 9, 10}, then the call of `collapse(list)` should return a new array containing {9, 17, 17, 8, 19}. The first pair from the original list is collapsed into 9 (7 + 2), the second pair is collapsed into 17 (8 + 9), and so on. If the list stores an odd number of elements, the final element is not collapsed. For example, if the list had been {1, 2, 3, 4, 5}, then the call would return {3, 7, 5}. Your method should not change the array that is passed as a parameter.

14. Write a method called `append` that accepts two integer arrays as parameters and returns a new array that contains the result of appending the second array's values at the end of the first array. For example, if arrays `list1` and `list2` store {2, 4, 6} and {1, 2, 3, 4, 5} respectively, the call of `append(list1, list2)` should return a new array containing {2, 4, 6, 1, 2, 3, 4, 5}. If the call instead had been `append(list2, list1)`, the method would return an array containing {1, 2, 3, 4, 5, 2, 4, 6}.

15. Write a method named gcd that accepts two integers as parameters and returns the greatest common divisor of the two numbers. The greatest common divisor (GCD) of two integers a and b is the largest integer that is a factor of both a and b. The GCD of any number and 1 is 1, and the GCD of any number and 0 is that number.

One efficient way to compute the GCD of two numbers is to use Euclid's algorithm, which states the following:

GCD(A, B) = GCD(B, A % B)
GCD(A, 0) = Absolute value of A

In other words, if you repeatedly mod A by B and then swap the two values, eventually B will store 0 and A will store the greatest common divisor.

For example: `gcd(24, 84)` returns 12, `gcd(105, 45)` returns 15, and `gcd(0, 8)` returns 8.

16. Write a method named `swapDigitPairs` that accepts an integer *n* as a parameter and returns a new integer whose value is similar to *n*'s but with each pair of digits swapped in order. For example, the call of `swapDigitPairs(482596)` would return `845269`. Notice that the 9 and 6 are swapped, as are the 2 and 5, and the 4 and 8. If the number contains an odd number of digits, leave the leftmost digit in its original place. For example, the call of `swapDigitPairs(1234567)` would return `1325476`. Do not use the String class to solve this problem

17.
```
public class ReferenceMystery3 {
    public static void main(String[] args) {
        int a = 7;
        int b = 9;
        Point p1 = new Point(2, 2);
        Point p2 = new Point(2, 2);
        addToXTwice(a, p1);
        System.out.println(a + " " + b + " " + p1.x + " " + p2.x);
        addToXTwice(b, p2);
        System.out.println(a + " " + b + " " + p1.x + " " + p2.x);
    }

    public static void addToXTwice(int a, Point p1) {
        a = a + a;
        p1.x = a;
        System.out.println(a + " " + p1.x);
    }
}
```

18. Write a class called BankAccount with following instance variables:

```
private String id;
private double balance;
private int transactions;
include constructor, accessors, mutators, equals, toString, deposit, withdraw, close. For each method
must make your own restrictions.
```

19. Create a class called rectangle. Come up with the list of the instance variables. Also add all the methods related to the rectangle class from practice-it to your class such as contain, unionRectangle, intersectionRectangle, accessors, mutators, toString, equals.

20. Create objects of the rectangle class and use all the methods from the rectangle class.

21. Write a class called Fractioon with the two instance variables denominator and numerator. Include constructors, toString, accessors, mutators, equals, also write the following methods:
Simplify that simplifies a given Fraction object, multiplication, subtraction, addition, division that can be used to multiply , subtract, divide, and add two objects of type fraction.