Exam 3 study Guide

Exam date:
T/Th lecture: Thursday May 12th,
Friday Lecture: Friday May 13th

Exam will cover chapters 5 and 7. You must have through knowledge of the previous chapters covered in class. This study guide contains some sample question for you to practice.

1. Convert the following `for` loops into an equivalent `while`

```
        for (int i = 1; i <= 2; i++) {
          for (int j = l ; j <= 3; j++) {
            for (int k = j; k <= 4; k++) {
              System.out.print("*");
            }
            System.out.print("!");
          }
          System.out.println();
        }
System.out.println();
```

2. Write a method called contains that examines a `String` and return whether it contains the given String. Returns true if it contains the string, and returns false otherwise. <u>You are not allowed to use the contains method from the String class. You solution must work with uppercase and lower case letters.</u> **Examples: contains ("Hello people in the World", "people") returns true, Contains ("Today is Thursday", "thur") returns true, Contains("I love Java programming", "hello") returns false**

3. What is the output of the following code

```
 public class Midterm3
{
   public static void main(String[] args)
   {


        System.out.println(mystery(12,18));
        System.out.println(mystery(0 , 10));
        System.out.println(mystery(-5, 10));

   }
   public static int mystery(int x, int y) {
    while (x != 0 && y != 0) {
        if (x < y) {
            y = y - x  ;
        } else {
            x = x - y  ;
        }
    }
    return x + y;
}
   }
```

4. Write a method named `printFactors` that accepts an integer as its parameter and uses a fencepost loop to print the factors of that number, separated by the word " and ". For example, the number 24's factors should print as:

```
1 and 2 and 3 and 4 and 6 and 8 and 12 and 24
```

You may assume that the number parameter's value is greater than 0.

5. 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`. You cannot solve this problem using a `String and the methods from the string class`.

6. **Write a method called compare that compares two arrays of String and returns true if the elements in both arrays are the same. Here is the signature for the method:**

Public boolean compare(string[] s1, String[] s2)

7. Self check Practice-it 7.17) 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`.

8. Self-check 7.19) The following program produces 4 lines of output. Write each line of output below as it would appear on the console.

```
import java.util.*;   // for Arrays class

public class ReferenceMystery1 {
    public static void main(String[] args) {
        int x = 0;
        int[] a = new int[4];
        x++;

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

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

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

9. Write a method called `median` that accepts an array of integers as its argument and returns the median of the numbers in the array. The median is the number that will appear in the middle if you arrange the elements in order. Assume that the array is of odd size (so that one sole element constitutes the median) and that the numbers in the array are between 0 and 99 inclusive.

For example, the median of `{5, 2, 4, 17, 55, 4, 3, 26, 18, 2, 17}` is `5`, and the median of `{42, 37, 1, 97, 1, 2, 7, 42, 3, 25, 89, 15, 10, 29, 27}` is `25`.

(*Hint*: You may wish to look at the `Tally` program from earlier in this chapter for ideas.)

10. Write a method `priceIsRight` that accepts an array of integers *bids* and an integer *price* as parameters. The method returns the element in the *bids* array that is closest in value to *price* without being larger than *price*. For example, if *bids* stores the elements`{200, 300, 250, 999, 40}`, then `priceIsRight(bids, 280)` should return 250, since 250 is the bid closest to 280 without going over 280. If all bids are larger than *price*, then your method should return -1.

The following table shows some calls to your method and their expected results:

Arrays	**Returned Value**
`int[] a1 = {900, 885, 989, 1};`	`priceIsRight(a1, 800)` returns 1
`int[] a2 = {200};`	`priceIsRight(a2, 120)` returns -1
`int[] a3 = {500, 300, 241, 99, 501};`	`priceIsRight(a3, 50)` returns -1

11. 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.

12. Write a method called pairs that accepts two arrays and list all the pairs . for example :
Int[] a = {1, 2 ,3};
Int[] b = {5,7,8};
Output: {(1,5),(1,7), (1,8), (2,5),(2,7),(2,8),(3,5),(3,7),(3,8)}