CS 11 – Mid-term exam #2

Name:_____

1. (20 points) Consider the short program below. What output does it produce?

```
class Foo {

    public static void main (String[] args) {

        int[] x = new int[4];
        int[] y = new int[4];

        for (int i = 0; i < x.length; i++) {

            x[i] = i;
            y[i] = i;

        }

        quux(x, y);

        System.out.println(x[1]);
        System.out.println(y[1]);

    } // main

    public static void quux (int[] a, int[] b) {

        if (a == b) {

            a = new int[6];
            a[1] = 15;
            b[1] = -15;

        } else {

            a = new int[2];
            a[1] = 100;
            b[1] = -100;

        }

        System.out.println(a.length);

    } // quux

} // class Foo
```

2. (20 points) Write a method that takes, as a parameter, a pointer to an array of `double` and returns nothing. This method should reverse the elements of the array. For example, if the array contains...

```
3.2  -1.5  0.0  9.58
```

Then after this method is complete, the array should contain...

```
9.58  0.0  -1.5  3.2
```

The method declaration is provided below as a starting point. Note that you should write this method such that it is *in-place*—that is, it should *not* create a new array as temporary working space, but rather it should use only the array passed by the caller. (Note that using additional variables is allowed, just not new arrays.)

```
public static void reverse (double[] x) {
```

3. (20 points) The worst case for *bubble sort* occurs when the original array has its smallest value at the end of the array (that is, at the highest numbered index). In this case, with each pass, that smallest value moves to the left by only one position, thus requiring all n passes before it reaches its final position.

   Consider a modified version of bubble sort, known as *bidirectional bubble sort*. For this algorithm, the passes over the array alternate direction instead of always progressing from left to right. Thus, in the example above, the very first right-to-left pass will bubble the smallest value down to index 0. Answer the following questions about this new algorithm:

   (a) What is the worst case input for bidirectional bubble sort? That is, what arrangement of initial values in an array will require the longest time for this algorithm to sort?

   (b) What is the *Big-O* running time for this algorithm? Support your claim by describing the number of compare and swap operations that the algorithm would perform for the worst case described above.

4. (20 points) Consider the following array of `int` values:

```
9  -2  3  8  1  0  2  -5
```

Now assume an implementation of the `mergeSort` method that prints two pieces of information every time it is called:

- As soon as it is called, it prints the contents of the array passed to it (that is, the unsorted input array).
- Just before it returns, it prints the contents of the array again (that is, the sorted result).

Write the output that would appear from calling `mergeSort` with the array given above.

5. (20 points) Write a method that accepts a pointer to an array of `int` and then randomly permutes the elements; that is, this method should shuffle the array into a random order.

```
public static void shuffle (int[] x) {
```