COSC–111 Introduction to Computer Science I
Practice for Midterm 02

1. Provide short answers (at most two sentences each) to the following questions.

1.a What does the instruction `int a = (int) (Math.random() * 10);` do?

A: It declares an `int` variable `a` and initializes it to a random number between 0 and 9 (inclusive).

1.b How do you create an array `a` of `doubles` with a length of 15?

A: `double[] a = new double[15];`

1.c What is the return type of the method with the following signature? How many parameters does it takes and what are their types?

```
public static double sum(int[] z; double start)
```

A: The method returns a `double`. It takes two parameters: the first is an array of `ints`, the second is a `double`.

2. What is the output when `ArrayPass` is run? What are the values of `q[6]` and of `q[9]`?

```
public class ArrayPass {
  public static void main (String[] args) {
    int[] q = new int[15];
    int i = 0;
    while (i < q.length) {
      q[i] = i * i;
      i = i + 1;
    }
    i = 5;
    moose(i, q);
    System.out.println(i);
    System.out.println(q[i]);
  }

  public static void moose (int i, int[] a) {
    i = i + 1;
    a[i] = a[i] * 2;
  }
}
```

A: The program prints

```
5
25
```

The value of `q[6]` is 72 and the value of `q[9]` is 81.

3. Write a method that compares two arrays of `int` to determine whether their contents are identical (i.e., the two arrays contain the same value in each position). It must return `true` if the contents are identical, and `false` otherwise. Its signature should be:
`public static boolean compare(int[] a, int[] b)`

A:

```
public static boolean compare(int[] a, int[] b) {
  if (a.length != b.length) {
    return false;
  }
  for (int i = 0; i < a.length; i++) {
    if (a[i] != b[i]) {
      return false;
    }
  }
  return true;
}
```

4. Write a variant `revInsertionSort` of `insertionSort` that sorts an array `a` of `int`s in *decreasing* order (i.e., after sorting, the first element of the array contains the *largest* value).

A:

```
public static void revInsertionSort(int[] a) {
  for (int i = 1; i < a.length; i++) {
    int t = a[i];
    int j = i-1;
    while (j >= 0 && t > a[j]) {
      a[j+1] = a[j];
      j--;
    }
    a[j+1] = t;
  }
}
```

The code is essentially the same as the original `insertionSort`, except that the second condition in the `while`-loop is now using > instead of <.

5. Modify the code for `merge` to sort the elements in decreasing order.
   A:

```
public static void merge(int[] a, int lo, int mid, int hi) {
  int i = lo;
  int j = mid + 1;
  int[] temp = new int[a.length];
  int temp_ctr = lo;
  while (temp_ctr <= hi) {
    if (i > mid || (j <= hi && a[j] > a[i])) {
      temp[temp_ctr] = a[j];
      j++;
      temp_ctr++;
    } else {
      temp[temp_ctr] = a[i];
      i++;
      temp_ctr++;
    }
  }

  for (temp_ctr = lo; temp_ctr <= hi; temp_ctr++) {
    a[temp_ctr] = temp[temp_ctr];
  }
}
```

The code is essentially the same as the original `merge`, except that the second condition in the `while`-loop is now checking whether `a[j] > a[i]`.