Mr. Brett Wortzman
Issaquah High School

Name: _____

Test 3 – Arrays

Section I – Multiple Choice

1. Which of the following is **true**?
   (A) The size of an array can be changed at any time.
   (B) The expression **arr.length** will return the size of an array called **arr**.
   (C) An array can contain values of multiple incompatible types at the same time.
   (D) The following is a valid Java statement that will create an array of four Strings:
   ```
   String[] stringArr = new String[“Hello”, “there”, “my”, “friend”];
   ```
   (E) After the following Java statement is executed, **arr** and **arr2** will contain the same elements:
   ```
   int[] arr2 = new int[arr.length];
   ```

2. Consider the following code segments:

   I.
   ```
   double [] array = new double [1028];
   ```
   II.
   ```
   double [] array;
   array = new double [1028];
   ```
   III.
   ```
   [] double array = new double [1028];
   ```

   Which code correctly initializes an array that holds 1,028 values of type **double**?
   (A) I only
   (B) II only
   (C) III only
   (D) I and II only
   (E) I, II, and III

3. Which of the following correctly initializes an array **superBass** to contain five elements each with value 0 ?

   I.
   ```
   int[] superBass = { 0, 0, 0, 0, 0 };
   ```
   II.
   ```
   int[] superBass = new int[5];
   ```
   III.
   ```
   int[] superBass = new int[5];
   for (int i = 0; i < superBass.length; i++) {
       superBass[i] = 0;
   }
   ```
   (A) I only
   (B) II only
   (C) I and II only
   (D) I and III only
   (E) I, II, and III

(Turn Over)

4. Which of the following code segments correctly declares a 2-by-3 two-dimensional?

   I. `int[][] twoDimensionalArray = new int[2, 3];`
   II. `int[][] twoDimensionalArray = new int[2][3];`
   III. `int[][] twoDimensionalArray = { {1, 2, 3}, {4, 5, 6} };`

   (A) I only
   (B) II only
   (C) III only
   (D) I and III only
   (E) II and III only

5. Consider the following code segment:

```
int[] numbers = new int [6];
numbers[1] = 4;
numbers[3] = 104;
numbers[5] = 2;

int x = numbers[1];
numbers[x] = 44;
numbers[numbers[5]] = 11;
```

   What are the values in the array **numbers** after this code segment has executed?

   (A) `{ 0, 44, 11, 104, 0, 2 }`
   (B) `{ 0, 4, 0, 104, 0, 2 }`
   (C) `{ 4, 0, 104, 0, 2, 0 }`
   (D) `{ 44, 11, 104, 0, 2, 0 }`
   (E) `{ 0, 4, 11, 104, 44, 2 }`

Questions 6-8 refer to the following static method.

```
public static int getA(int[] a, int b) {
    return a[b];
}
```

6. What is the value of **y** after the following code executes?

```
int[] x = { 1, 2, 3, 4, 5 };
int y = getA(x, 1);
y = getA(x, y);
```

   (A) 1
   (B) 3
   (C) 5
   (D) An exception is thrown because an array index is less than zero
   (E) An exception is thrown because an array index is too large

7. What is the value of **y** after the following code executes? (See previous page for the definition of the static method **getA**).

```
int[] x = { 1, -1, 5, 1, 3};
int y = getA(x, 1);
y = getA(x, y);
```

(A) 1
(B) 3
(C) 5
(D) An exception is thrown because an array index is less than zero
(E) An exception is thrown because an array index is too large

8. What is the value of **y** after the following code executes? (See previous page for the definition of the static method **getA**).

```
int[] x = { 5, 4, 3, 2, 1 };
int y = getA(x, getA(x, 1));
```

(A) 1
(B) 3
(C) 5
(D) An exception is thrown because an array index is less than zero
(E) An exception is thrown because an array index is too large

9. Consider the following Java program:

```
public class ChopItUp {
    public static String[] chop(String input) {
        String[] output = new String[input.length()];
        for (int i = 0; i < input.length(); i++) {
                output[i] = input.substring(i);
        }
        return output;
    }

    public static void main(String[] args) {
        String[] what = chop("Hello, world.");
        System.out.println(what[2]);
    }
}
```

What is output to the console when this program is executed?

(A) Hello, world.
(B) llo, world.
(C) Hello, worl
(D) Hello,
(E) world.

10. Consider the following Java program:

```
public class ReferenceMystery1 {
    public static void main(String[] args) {
        int x = 0;
        int[] a = new int[8];
        x = x + 2;
        mystery(x, a);

        x = x + 2;
        mystery(x, a);
    }

    public static void mystery(int x, int[] a) {
        x = x + 1;
        a[x] = a[x] + 1;
    }
}
```

What are the values in the array **a** after the **main** method has finished?

(A) `{ 0, 0, 0, 1, 0, 1, 0, 0 }`
(B) `{ 0, 0, 0, 0, 0, 0, 0, 0 }`
(C) `{ 0, 0, 0, 1, 0, 0, 0, 1 }`
(D) `{ 0, 0, 1, 0, 1, 0, 0, 0 }`
(E) `{ 0, 0, 1, 1, 0, 0, 0, 0 }`

Section II – Free Response

11. For each of the following code samples, assume the code is intended to do what the comment describes. Identify the bug in each sample that causes it not to work as intended, and propose a fix. In all code samples, assume referenced arrays and lists have already been declared and initialized.

    a.
```
// output the sum of each pair of neighboring elements in array arr
for (int i = 0; i < arr.length; i++) {
     System.out.println(arr[i] + arr[i+1]);
}
```

    b.
```
// output the elements in array arr in reverse order
for (int i = arr.length; i > 0; i--) {
     System.out.println(arr[i]);
}
```

    c.
```
// output the sum of all the elements of a two-dimensional array arr
int sum = 0;
for (int i = 0; i < arr.length; i++) {
     for (int j = 0; j < arr.length; j++) {
          sum += arr[i][j];
     }
}
System.out.println(sum);
```

    d.
```
// copy all the elements of two-dimensional array arr into arr2
for (int i = 0; i < arr.length; i++) {
     for (int j = 0; j < arr[i].length; i++) {
          arr2[i][j] = arr[i][j];
     }
}
```

    e.
```
// put the numbers 1 through 10 in an ArrayList list
for (int i = 1; i <= 10; i++) {
    list[i – 1] = i;
}
```

12. Write a Java method called **`fillWithSquares`** that takes an integer **`n`** as an argument and returns a new array filled with the first **`n`** perfect squares (1, 4, 9, 16, etc.).

13. Write a Java method called **`findLongest`** that takes an array of Strings and returns the longest String in the array. If there is a tie, return the String the appears first in the array. For example, if **`findLongest`** is passed the array **`{"abc", "defg", "hi", "j", "kl"}`** it should return **`"defg"`**. If **`findLongest`** is passed the array **`{"ab", "c", "de"}`** it should return **`"ab"`**.

14. Write a Java method called **`reverseList`** that takes an **`ArrayList`** of **`String`**s as an argument and returns a new list containing the same **`String`**s but in reverse order.

15. Write a Java method called **`sumRows`** that takes a two-dimensional array of **`double`**s as an argument and returns a new array containing the sum of the elements in each row of the argument array. So, for example, if the argument is: **`{ {1, 3, 5}, {2, 4, 6}, {0, 0, 1} }`** then the return value should be **`{9, 12, 1}`**.