Name (print): ________________________________

• INSTRUCTIONS:
  – Keep your eyes on your own paper and do your best to prevent anyone else from seeing your work.
  – Do NOT communicate with anyone other than the professor/proctor for ANY reason in ANY language in ANY manner.
  – This exam is closed notes, closed books, and no calculator.
  – Turn all mobile devices off and put them away now. You cannot have them on your desk.
  – Write neatly and clearly indicate your answers. What I cannot read, I will assume to be incorrect.
  – Stop writing when told to do so at the end of the exam. I will take 5 points off your exam if I have to tell you multiple times.
  – Academic misconduct will not be tolerated. Suspected academic misconduct will be immediately referred to the Emory Honor Council. Penalties for misconduct will be a zero on this exam, an F grade in the course, and/or other disciplinary action that may be applied by the Emory Honor Council.

• TIME: This exam has 7 questions on 11 pages including the title page. Please check to make sure all pages are included. You will have 50 minutes to complete this exam.

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Emory community. I have also read and understand the requirements and policies outlined above.

Signature: ________________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>14</td>
<td>30</td>
<td>4</td>
<td>16</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>100</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. (14 points) Variable types and kinds. Consider the following code:

```java
public class Code1 {
    static int first = 0;
    public static void mysteriousMethod(int a, int b, int[] c) {
        first = c[1]++ - first;
        int temp = b;
        a = ++temp;
        c[0] = 4;
        System.out.println(a);
    }
    public static double[] doSomething(int d) {
        d++;
        return new double[d];
    }
    static double second = 1;
    public static void main(String[] args) {
        int x = 100, y = 200;
        int[] z = {300, 400};

        System.out.println(x + " " + y + " " + z[0] + " " + z[1]);
        System.out.println(first + " " + second);
        mysteriousMethod(x, y, z);
        System.out.println(x + " " + y + " " + z[0] + " " + z[1]);
        System.out.println(first + " " + second);
        double[] arr = doSomething(z[0]);
        System.out.println(z[0] + " " + z[1]);
        System.out.println(arr.length);
    }
}
```

Solution

1. List all parameter variables in the `mysteriousMethod` method. (3 points)
   int a, int b, int[] c

2. List all class variables in the program above. (2 points)
   int first, double second

3. What would be the output of the program? (9 points)
   
   ```
   100 200 300 400 (2 pts)
   0 1.0
   201
   100 200 4 401 (2 pts)
   400 1.0
   4 401
   5
   ```
2. (30 points) For each code fragment below, provide the output of the code. If the code contains errors, identify **ALL ERRORS** in that code fragment.

Assume that the method `Arrays.toString(arrayName)` will return a String containing the elements of the parameter array, where elements are separated by commas. The Java library method `Arrays.toString( params )` are overloaded so it can accept a parameter array of any type (`char[]`, `int[]`, `double[]`, `boolean[]` and etc).

E.g.
```java
int[] x = {12, 23, 54, 11};
System.out.println(Arrays.toString(x));
```
will output `12, 23, 54, 11` to the terminal.

a) Code:
```java
public class Code2A {
    public static void main(String[] args) {
        int[] arr = {1, 2, 0, 1, 9};
        for (int idx = 2; idx < arr.length; idx++) {
            if (arr[idx] <= arr[idx + 1]) {
                System.out.println(idx);
            }
        }
        System.out.println(Arrays.toString(arr));
    }
}
```
Output: Error
2
3
`ArrayIndexOutOfBoundsException:`
due to `arr[idx + 1]` evaluates to `arr[5]`

b) Code:
```java
public class Code2B {
    public static void main(String[] args) {
        int[] c = {5, 6, 9, 8};
        System.out.println(Arrays.toString(c));

        double[] d = {2.0, 4.0, 6.0};
        for (int k = d.length - 1; k > 0; k--) {
            c[k] = (int) d[k];
        }

        System.out.println(Arrays.toString(d));
        System.out.println(Arrays.toString(c));
    }
}
```
Output:
`5, 6, 9, 8`
2.0, 4.0, 6.0
5, 4, 6, 8

c) Code:

```java
public class Code2C {
    public static void main(String[] args) {
        int[] a = {1, 4, 7};
        int[] b = new int[3];
        int[] c = new int[5];
        c = a;
        a = b;
        b[2] = c[1];

        System.out.println(a.length);
        System.out.println(Arrays.toString(b));
        System.out.println(Arrays.toString(c));
    }
}
```

Output:
3
0, 0, 4
1, 4, 7
d) Code:

```java
public class Code2D {
    public static void main(String[] args) {
        int[] b = {2, 5, 7};
        int[][] a = new int[4][5];
        a[1] = b;
        b = new int[2];
        System.out.println(a[1][2]);
        System.out.println(b[0]);
    }
}
```

Output:
7
0

---

e) Code:

```java
public class Code2E {
    public static void main(String[] c) {
        boolean first = callMethod(4);
        System.out.println(first);
    }

    public static int callMethod(boolean d) {
        if (d < 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

Output: Error (cannot compile)
callMethod: mismatch between parameter types and also return types
3. (4 points) Algorithms:
Assume, as in question 2, that the method Arrays.toString( arrayName) will return a String containing
array elements delimited/separated by commas.
Consider the following 2 arrays:
int[] a = {15, 23, 42, 51, 31, 88, 92, 100};
int[] b = {-10, 11, 18, 17, 25, 29};

a) (2 pts) For each array a and b, can the linear (sequential) search algorithm be performed on
the array? If yes, list the elements in the order we will inspect when searching for value 21. If
not, explain why.

Yes for both a and b. For a, we will examine: 15, 23, 42, 51, 31, 88, 92, 100 (sequentially from
left to right).
For b, we will examine: -10, 11, 18, 17, 25, 29 (sequentially from left to right).

b) (2 pts) For each array a and b, can the binary search algorithm be performed on the array? Why
or why not?

No for neither a nor b, since neither array is sorted.
4. (16 points) Method overloading and variable scopes.

What is the output of following Java program?

```java
public class Scope {
    public static int z = 5;

    public static void main(String[] args) {
        
        System.out.println(z);
        boolean z = true;
        System.out.println(z + " = " + Scope.z + "?");
    }
    System.out.println(z);
    z = z(z);

    System.out.println(z);
    z = 10;
    for (int z = 10; z < 1000; z += 100) {
        if (z % 200 == 10) {
            System.out.println(z);
        }
    }
    System.out.println(z);
    System.out.println(method1(z));
    System.out.println(z);
}

public static String z(String z) {
    return z + z;
}

public static int z(int z) {
    return z * z;
}

public static int method1(int z) {
    System.out.println("in method1");
    z = method2(z);
    Scope.z++;
    System.out.println("almost done with method1");
    return z;
}

public static int method2(int z) {
    System.out.println("in method2");
    return z + 1 / z;
}

5
true = 5?
5
25
5. (10 points) Methods (Write both header and method body)
Write a complete method named `reorganize` that accepts an array of integers, and returns a new array containing all even elements, followed by all odd elements from the original array.

Example:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>reorganize({2, 7, 9, 8, 3, 1})</code></td>
<td><code>{2, 8, 7, 9, 3, 1}</code></td>
</tr>
<tr>
<td><code>reorganize({1, 2, 3, 4, 5, 6})</code></td>
<td><code>{2, 4, 6, 1, 3, 5}</code></td>
</tr>
<tr>
<td><code>reorganize({0, 8, 9, 4})</code></td>
<td><code>{0, 8, 4, 9}</code></td>
</tr>
<tr>
<td><code>reorganize({7, 3, 1})</code></td>
<td><code>{7, 3, 1}</code></td>
</tr>
<tr>
<td><code>reorganize({10, 12, 8, 7, 1, 2})</code></td>
<td><code>{10, 12, 8, 2, 7, 1}</code></td>
</tr>
</tbody>
</table>

```java
public static int[] reorganize(int[] arr) {
    int[] result = new int[arr.length];
    int idx = 0;

    // "insert" all even numbers in the result array
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] % 2 == 0) {
            result[idx] = arr[i];
            idx++;
        }
    }

    // "insert" all odd numbers in the result array
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] % 2 == 1) {
            result[idx] = arr[i];
            idx++;
        }
    }

    return result;
}
```
6. (12 points) Methods (Write both the header and method body)

Write a complete method named `resizeArray` that accepts an array of ints `arr`, and a parameter of type int `newSize`.

If `newSize` is larger than or equal to the length of the array `arr`, "expand" the array, i.e. the method should return a new array with length `newSize` containing all elements of `arr` in the original order. You can leave 0s in the remaining unfilled slots of the array.

If `newSize` is smaller than the length of the array, "reduce" the array, i.e. the method should return a new array with length `newSize` containing only the first `newSize` elements of the original array `arr`.

For example:

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Returned array</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>resizeArray({ 1, 2, 3, 4, 5}, 3)</code></td>
<td>{1, 2, 3}</td>
</tr>
<tr>
<td><code>resizeArray({2, 9}, 6)</code></td>
<td>{2, 9, 6, 0, 0, 0}</td>
</tr>
<tr>
<td><code>resizeArray({1, 3}, 5)</code></td>
<td>{1, 3, 0, 0, 0}</td>
</tr>
<tr>
<td><code>resizeArray({21, 5, 9, 17, 11}, 2)</code></td>
<td>{21, 5}</td>
</tr>
</tbody>
</table>

**Solution:**

```java
public static int[] resizeArray(int[] arr, int newSize) {
    int[] result = new int[newSize];
    if (newSize >= arr.length) {
        for (int i = 0; i < arr.length; i++) {
            result[i] = arr[i];
        }
    } else {
        for (int i = 0; i < newSize; i++) {
            result[i] = arr[i];
        }
    }
    return result;
}
```

Algorithm: 4 pts
Method header: 2 pts
Create a result array: 1 pt
If-statement: 1 pt
For-loop: 2 pts
Return statement: 1 pt
Correct syntax: 1 pt
7. (14 points) 2D array

Write a complete method named `getProducts` that accepts a rectangular two-dimensional array of ints and also a boolean parameter `useColumns`.

If `useColumns` is true, the method should return a new array containing the products of elements in each column.

If `useColumns` is false, the method should return a new array containing the products of elements in each row.

You can assume that the parameter array is a "rectangular" array.

Example:

If `useColumns` is true and the array is: 
```
{ { 1, 2, 3, 4 },
  { 2, 5, 1, 7 } }
```
the method should return `{ 1 * 2, 2 * 5, 3 * 1, 4 * 7}`,
which is `{ 2, 10, 3, 28}`

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Returned array</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getProducts({{ 1, 2, 3, 4 }, { 2, 5, 1, 7 }}, true)</code></td>
<td><code>{2, 10, 3, 28}</code></td>
</tr>
<tr>
<td><code>getProducts({{ 2, 9, 4, 0 }, { 4, 3, 2, 1 }}, false)</code></td>
<td><code>{0, 24}</code></td>
</tr>
<tr>
<td><code>getProducts({{ 5, 2, 3 }, { -4, 1, 4 }}, false)</code></td>
<td><code>{30, -16}</code></td>
</tr>
<tr>
<td><code>getProducts({{ 6, 4, 2 }, {3, 0, 1}}, true)</code></td>
<td><code>{18, 0, 2}</code></td>
</tr>
</tbody>
</table>

Solution:

```java
public static int[] getProducts(int[][] arr, boolean useColumns) {
    if (useColumns) {
        int[] result = new int[arr[0].length];
        for (int col = 0; col < arr[0].length; col++) {
            result[col] = 1;
            for (int row = 0; row < arr.length; row++) {
                result[col] *= arr[row][col];
            }
        }
        return result;
    } else {
        int[] result = new int[arr.length];
        for (int row = 0; row < arr.length; row++) {
            result[row] = 1;
            for (int col = 0; col < arr[0].length; col++) {
                result[row] *= arr[row][col];
            }
        }
        return result;
    }
}
```

Algorithm: 4 pts
Header: 3 pts
Nested for-loops: 3 pts
Creating result array of correct size: 1 pt
Correct product computation: 2 pts
Return statement: 1 pt

10
### ASCII Table

<table>
<thead>
<tr>
<th>0 NUL</th>
<th>1 SOH</th>
<th>2 STX</th>
<th>3 ETX</th>
<th>4 EOT</th>
<th>5 END</th>
<th>6 ACK</th>
<th>7 BEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS</td>
<td>HT</td>
<td>NL</td>
<td>VT</td>
<td>MP</td>
<td>CR</td>
<td>SO</td>
<td>ST</td>
</tr>
<tr>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>DC4</td>
<td>MAK</td>
<td>SYN</td>
<td>ETB</td>
<td></td>
</tr>
<tr>
<td>CAN</td>
<td>EM</td>
<td>SUB</td>
<td>ESC</td>
<td>FS</td>
<td>GS</td>
<td>RS</td>
<td>US</td>
</tr>
<tr>
<td>SP</td>
<td>I</td>
<td>&quot;</td>
<td>#</td>
<td>&amp;</td>
<td>%</td>
<td>‘</td>
<td>,</td>
</tr>
<tr>
<td>(</td>
<td>)</td>
<td>+</td>
<td>=</td>
<td>&gt;</td>
<td>-</td>
<td>.</td>
<td>/</td>
</tr>
<tr>
<td>@</td>
<td>#</td>
<td>$</td>
<td>&amp;</td>
<td>(</td>
<td>)</td>
<td>_</td>
<td>`</td>
</tr>
<tr>
<td>64 H</td>
<td>65 A</td>
<td>66 B</td>
<td>67 C</td>
<td>68 D</td>
<td>69 E</td>
<td>70 F</td>
<td>71 G</td>
</tr>
<tr>
<td>72 H</td>
<td>73 I</td>
<td>74 J</td>
<td>75 K</td>
<td>76 L</td>
<td>77 M</td>
<td>78 N</td>
<td>79 O</td>
</tr>
<tr>
<td>80 P</td>
<td>81 Q</td>
<td>82 R</td>
<td>83 S</td>
<td>84 T</td>
<td>85 U</td>
<td>86 V</td>
<td>87 W</td>
</tr>
<tr>
<td>88 X</td>
<td>89 Y</td>
<td>90 Z</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>96</td>
<td>97 a</td>
<td>98 b</td>
<td>99 c</td>
<td>100 d</td>
<td>101 e</td>
<td>102 f</td>
<td>103 g</td>
</tr>
<tr>
<td>104 h</td>
<td>105 i</td>
<td>106 j</td>
<td>107 k</td>
<td>108 l</td>
<td>109 m</td>
<td>110 n</td>
<td>111 o</td>
</tr>
<tr>
<td>112 p</td>
<td>113 q</td>
<td>114 r</td>
<td>115 s</td>
<td>116 t</td>
<td>117 u</td>
<td>118 v</td>
<td>119 w</td>
</tr>
<tr>
<td>120 x</td>
<td>121 y</td>
<td>122 z</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
<td>127 DEL</td>
</tr>
</tbody>
</table>

Note that uppercase: $65 \leq x \leq 90$  
Difference between A (65) and a (97) is 32!  
Note that lowercase $97 \leq x \leq 122$  
Difference between Q (81) and q (113) is 32!