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 12 pages including the title page. Please check to make sure all pages are included. You will have 75 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>9</td>
<td>6</td>
<td>10</td>
<td>18</td>
<td>9</td>
<td>10</td>
<td>13</td>
<td>75</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. (9 points) Each question below contains at 2 different errors. Find the errors and circle them. Select one of the errors, mark it with an (*) and correct it.

(a) // This code should print out "Didn’t wake up on time."
   // as many times as false occurs in the array.
   boolean[] wokeUpOnTime = {true, true, false, true, false};
   for(i = 0; i < wokeUpOnTime.length; i++){
       if(wokeUpOnTime[i] = false){
           System.out.println("Didn’t wake up on time.");
       }
   }

(b) // This code should fill the end of an array with values and
    // print the element that occurs in the middle of the array.
    int x = 5, y = 3, z = 2;
    int[] xyz = new int[5];
    xyz[3] = x;
    xyz[4] = y;
    xyz[5] = z;
    System.out.println("Middle element is: " + xyz[xyz.length%2]);

(c) //This code should print a greeting in reverse.
    String greeting = "hello!";
    for(int i = greeting.length(); i > 0; i++){
        System.out.print(greeting[i]);
    }
    System.out.println(); //make it look prettier
2. Assume you are given the initial array:
   ```
   int[] a = {9, 7, 16, 30, 8, 4};
   ```
   For each of the given algorithms below, draw the array after 1 element has been sorted. You **DO NOT** need to draw the entire sorting process. Simply draw the array after one element has been sorted. Begin each algorithm with the initial array above.
   
   (a) (2 points) Selection Sort

   (b) (2 points) Insertion Sort

   (c) (2 points) Bubble Sort
3. (points) Consider the following code:

```java
public class Exam{
    public static double minScore = 70.0; //position (1)

    public static void addPoints(double minScore, double points){
        minScore += points;
    }

    public static void addPoints(double[] minScore, double points){
        for(int i=0; i<minScore.length; i++) {
            minScore[i] += points;
        }
    }

    public static boolean isFailed(double minScore){ //position (2)
        return minScore < Exam.minScore;
    }

    public static boolean isFailed(double[] scores){
        double avg = 0.0;
        int count = 0;
        for(int i = 0; i < scores.length; i++){
            int minScore = 0;
            avg = avg + scores[i];
            if( minScore < scores[i] ) //position (3)
                count++;
        }
        avg = avg / scores.length;
        return avg < minScore; //position (4)
    }

    public static void main(String[] args){
        double[] rawScores = {60.0, 75.0, 85.0};
        System.out.println(minScore);
        addPoints(rawScores, 5);
        double minScore = rawScores[0]; //position (5)
        System.out.println(minScore);
        if(isFailed(rawScores) == true)
            System.out.println("Fail!");
        else
            System.out.println("Pass!");
        addPoints(rawScores[0], 10);
        minScore=rawScores[0];
        System.out.println(minScore);
        if(isFailed(rawScores[0]))
            System.out.println("Fail!");
        else
            System.out.println("Pass!");
    }
}
```
(a) (5 points) At the following positions (identified by a comment in the code), identify whether minScore is a local, class, or parameter variable.
   i. position(1)

   ii. position(2)

   iii. position(3)

   iv. position(4)

   v. position(5)

(b) (5 points) What is the output of this program?
4. For each of the following code fragments give the output. If the fragment would result in an infinite loop, write “infinite loop” and the first 3 outputs.

(a) (2 points) Assume the program is invoked with the command line:
    java Program 3 15.5 4
    public static void main(String[] args) {
        System.out.println(args[1] + args[0] + args[2]);
    }

(b) (2 points)
    int month = 3;
    switch (month) {
        case 1:
            System.out.println("Jan");
            System.out.println("MLK, Jr Day");
        case 2:
            System.out.println("Feb");
            System.out.println("Valentine’s Day");
        case 3:
            System.out.println("Mar");
            System.out.println("It’s Spring Break");
        case 4:
            System.out.println("Apr");
            System.out.println("Final Exams");
            break;
        default:
            System.out.println("Invalid month");
    }

(c) (2 points) int x = 0;
    for(int x = 0; x < 8; x++) {
        if (x % 2 == 0) {
            continue;
        }
        System.out.println(x+1);
    }
(d) (3 points)
for (int i = 0; i < 12; i++) {
    System.out.println(i);
    if (i >= 7) {
        break;
    } else {
        i = i + 1;
    }

}

(e) (3 points)
int i = 9;
while (i > 2) {
    if (i % 3 == 0) {
        i = i - 1;
    } else if (i % 4 == 1) {
        i = i - 2;
    } else {
        i = i - 3;
    }
    System.out.println(i);
}
(f) (6 points)

```java
int[][] g = {{3, 9, -11},
            {15, 4, 0, -2},
            {-9, 16, -8}};

int x = g[0][0];
int y = g[2][2];
int z = 0;

for (int i = 0; i < g.length; i++) {
    for (int j = 0; j < g[i].length; j++) {
        if (x > g[i][j]) {
            x = g[i][j];
        }
        if (y < g[i][j]) {
            y = g[i][j];
        }
        if (g[i][j] % 2 == 0) {
            z++;
        }
    }
}

System.out.println("x: " + x);
System.out.println("y: " + y);
System.out.println("z: " + z);
```
5. Assume you are given the array:
   ```java
   int[] a = {8, 10, -2, 5, 11};
   ```
   and the following methods:

   ```java
   public static void method1(int[] array) {
       for (int i = 0; i < array.length/2; i++) {
           int temp = array[i];
           array[ array.length - 1 - i ] = array[i];
           array[i] = temp;
       }
   }
   public static void method2(int x, int y) {
       int temp = x;
       x = y;
       y = temp;
   }
   public static void method3(int[] array) {
       array = new int[4];
       array[0] = 1;
       array[1] = 3;
       array[2] = 5;
       array[1] = array[2];
       array[3] = array[0];
   }
   ```

   Draw the array `a` after each of the following method calls. Assume the method calls are independent; in other words, always begin with the initial array above.

   (a) (3 points) method1(a)

   (b) (3 points) method2(a[0], a[3])

   (c) (3 points) method3(a)
6. (10 points) Write a method, `zipStrings`, that takes two strings as parameters, “zips” them together, and returns the resulting string. To “zip” two strings, create a new string made of the first char of the first string, then the first char of the second, then the second char of first string, the second char of second string, and so on. Any leftover chars go at the end of the result. One or both strings may be the empty string. Example inputs and return values:

```java
zipStrings("abc", "b") ⇒ "abbc"
zipStrings("Valerie", "Summet") ⇒ "VSaulmemreite"
zipStrings("", ") ⇒ ""
zipStrings("Aaa", "bBb") ⇒ "AbaBab"
```
7. There are 2 parts to this question. For part (b), you can use the method described in part (a) even if you do not write code for it. Moreover, you may assume any code you write for part (a) functions correctly and as described.

(a) (8 points) Write a method named `minAfter` which takes two parameters: an array of integers and an integer which represents an index in that array. Your function should return the index of the smallest element in the array which occurs at or after the index position given by the integer parameter. Examples:

`minAfter({5, 2, 4, -1}, 0)` returns 3 since the smallest element that occurs at or after index 0 is -1.

`minAfter({-1, 2, 5, 3}, 1)` returns 1 since the smallest element that occurs at or after index 1 is 2.
(b) (5 points) Write a code fragment (doesn’t have to be a program or a method) which would implement selection sort for an array of integers, $a$. In other words, after your code runs, the array $a$ should be in sorted order from least to greatest values. **Hint:** use the method from part (a)!