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 10 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>9</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

Score: ____________________________
1. (9 points) Each question below contains 2 different syntax, runtime, or logic errors. Find them in each question, and circle what's causing the error. Pick one of the errors for each question, put a * next to it (indicating your choice), and explain why it's an error. You do not need to indicate which of the three types of errors it is.

(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 an array with values, print it,  
    // and change the values. 
    int x = 5, y = 3, z = 2;  
    int[] xyz = new int[3];  
    xyz[1] = x;  
    xyz[2] = y;  
    xyz[3] = z;  
    for(int[] element : xyz){  
        System.out.println("Original element is " + element); 
    } 
    int i = 0;  
    do{  
        if(x >= 0){  
            x -= 2;  
        }  
        i++;  
    } while(i < 3);
(c) //This code should print a greeting in reverse.
   String greeting = "hello!";
   for(int i = greeting.length; i >= 0; i++){
       System.out.print(greeting.charAt(i));
   }
   System.out.println(); //make it look prettier
2. (9 points) For each of the code fragments below, write in the whitespace below the code what output is produced when `bar()` is called. Take care to format the output correctly.

(a) public static double foo(int val){
    System.out.print("beep! ");
    double out = val/2;
    if(val > 0){
        val = val + 2;
        out = out - 2;
    }
    return out;
}

public static void bar(){
    int val = 5;
    int x = 3;
    double y;
    
    x = (int) foo(val);
    y = foo(x);
    System.out.println("val has value " + val);
    System.out.println("x has value " + x);
    System.out.println("y has value " + (int) (y++));
}

(b) public static void foo(int[] array) {
    for(int i = 0; i < array.length; i++) {
        if (i == (array.length - 2)) {
            continue;
        }
        array[i] = i;
    }
    System.out.println("Beep boop!");
}

public static void bar() {
    int[] values = {1, 2, 3, 4};
    int[] other = values;
    foo(values);
    other[0] = -4;

}
3. For each of the following questions, consider the series of arrays as steps in a sorting algorithm iteration. Identify on the line provided which algorithm is being performed, and write the array as it would appear after the next iteration in the whitespace provided.

(a) (2 points) Sorting algorithm: ________________
   {0, 4, 1, 3, 6} (original array)
   {0, 4, 1, 3, 6}
   {0, 1, 4, 3, 6}

(b) (2 points) Sorting algorithm: ________________
   {5, 3, -1, 2, 0} (original array)
   {-1, 3, 5, 2, 0}
   {-1, 0, 5, 2, 3}

(c) (2 points) Sorting algorithm: ________________
   {9, 8, 7, 6, 5} (original array)
   {5, 8, 7, 6, 9}
4. For the following arrays, list the number of values inspected (so, how many values are checked before the answer is found) for the given searching algorithm. For example, if the first element that the search checks is the key, the number of elements inspected is 1. Also list the index that’s returned by the search. If the element is not found, assume the algorithm returns -1. If the algorithm is not applicable, write “impossible” instead.

(a) (1 point) \{0, 2, 3, 1, 5\} linear search with key 5.

Number of elements inspected:

Index returned:

(b) (1 point) \{0, 2, 1, 3, 5, 6, 7\} binary search with key 6.

Number of elements inspected:

Index returned:

(c) (1 point) \{-2, 1, 3, 5, 9, 11, 20\} binary search with key 21.

Number of elements inspected:

Index returned:

(d) (1 point) \{0, 2, 3, 1, 5, 9, 11, 20\} linear search with key 10.

Number of elements inspected:

Index returned:
5. (6 points) Write a method, `hasThreeInARow`, that takes in an array of integers and returns a boolean which is true if there are three equal numbers in a row in the array, and false otherwise. **The array may be empty.**
Example input arrays → return values:
- `{}` → false
- `{1, 2, 3, 4, 3, 3}` → false
- `{1, 2, 2, 2}` → true
6. (6 points) Write a method, `zipStrings`, that takes two strings and “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 → outputs:

- ("aaa", "b") → "abaab"
- ("kiwi", "bird") → "kbiwrid"
- ("", ") → ""
- ("Aaa", "bBb") → "AbaBab"
7. (10 points) Given that a method with the method header

    public static void moveUp(String[] array, int index)

already exists, write a method overloading the moveUp method. The overloaded method
should move an element in an integer array (instead of String) at the specified index
up (closer to the beginning of the array), displacing other elements. Your overloaded
method should take an additional parameter, an integer number that indicates how far
up the element should be moved. This value is guaranteed to be at least one, and the
resulting position is guaranteed to be valid. No value should be returned.

Example parameters → resulting values in the integer array (note: not return values):
(new int[] {7, 8, 9}, 2, 2) → {9, 7, 8}
(new int[] {7, 8, 9}, 2, 1) → {7, 9, 8}
(new int[] {7, 8, 9}, 1, 1) → {8, 7, 9}