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 6 questions on 8 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>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

Score:  .................................................................
1. Consider the function below which is part of a Java program. Assume that sNums is always a sorted array.

```java
public static int myMethod(int[] sNums, int n){
    int begin = 0, end = sNums.length-1;
    while (begin <= end){
        int mid = begin + (end-begin) / 2;
        if(sNums[mid] == n){
            return mid;
        } else if(sNums[mid] < n){
            begin = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return -1;
}
```

Answers the following questions.

(a) (1 point) What type is the return type of the function `myMethod`? int

(b) (1 point) How many local variables does the `myMethod` method have and what is/are their names? 3; begin, end, mid

(c) (1 point) How many parameter variables does the function `myMethod` have and what is/are their names? 2; sNums and n

(d) (4 points) What will this method return for the following calls?

- `what({5,9,12,30,76,100,125}, 100)` --> 5
- `what({5,9,12,30,76,100,125}, 8)` --> -1
- `what({5,9,12,30,76,100,125}, 12)` --> 2

(e) (3 points) What is this method accomplishing? It searches for a given number in a sorted array and returns the index of the number if it is found, otherwise returns -1. This algorithm is called binary search which in each step, compares the given number with the middle number and then shrinks the search elements to the right or left side of that element.
2. (6 points) Write the output that the following code will print. You may assume the code compiles as written.

```java
public class Question2 {
    public static int k = 1;
    public static boolean b = false;

    public static boolean compareK(int k) {
        if (Question2.k < k) {
            Question2.k = k;
            return !b;
        }
        return b;
    }

    public static void addK(int[] nums, int k) {
        System.out.println(k);
        for (int i = 0; i < nums.length; i++)
            nums[i] = nums[i] + k;
        k = 3;
    }

    public static void main(String[] args) {
        System.out.println(k);
        int k = 2;
        int[] arr1 = {1, 2, 3, 4, 5};
        System.out.println(Arrays.toString(arr1));
        addK(arr1, k);
        System.out.println(Arrays.toString(arr1));
        System.out.println(k);
        boolean result = compareK(k);
        System.out.println(Question2.k);
    }
}
```

```
1
[1, 2, 3, 4, 5]
2
[3, 4, 5, 6, 7]
2
2
```
3. (7 points) For each entry below, state whether the code is correct as written or has an error. Show where the error is (you can circle the incorrect code) and explain why it is an error.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error?</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static double foo(double[][] nums) {</td>
<td>No</td>
<td>it will return the last element in the first row</td>
</tr>
<tr>
<td>return nums[0][nums[0].length-1] + 1;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>String[] s = {&quot;1&quot;,&quot;2&quot;,&quot;3&quot;};</td>
<td>Yes</td>
<td>s[1] is the string &quot;2&quot; and its second char doesn’t exist</td>
</tr>
<tr>
<td>System.out.println(s[1].charAt(1));</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boolean a = true;</td>
<td>Yes</td>
<td>b is declared inside if statement Thus b++ will be using an out of scope variable</td>
</tr>
<tr>
<td>if(a &amp;&amp; false){</td>
<td></td>
<td></td>
</tr>
<tr>
<td>int b = 2;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b ++;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>int[][] a;</td>
<td>Yes</td>
<td>a is not initialized with a size</td>
</tr>
<tr>
<td>a[0][0] = 5;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public static boolean foo(int num) {</td>
<td>Yes</td>
<td>The method should return in all cases but it returns nothing for a greater or equal to 5</td>
</tr>
<tr>
<td>if( num &lt; 5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return false;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Consider the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>What?</th>
</tr>
</thead>
</table>
| public static int methodA(int[] nums, int n){
  if(n < nums.length && n>=0)
    return nums[n];
  else
    return -1;
} | returning the array element in the given index n if n out of bound, returning -1 |
| public static int methodB(int[] nums, int n){
  int i = 0;
  while(i<nums.length && nums[i]!=n){
    i++;
  }
  if(i < nums.length)
    return i;
  return -1;
} | finding the given number in the array and returning its index if doesn’t exist, returning -1 |
| public static int methodC(int[] nums, int n){
  int k = -1;
  for(int i=0;i<nums.length;i++){
    if(nums[i] == n){
      k = i;
      break;
    }
  }
  return k;
} | finding the given number in the array and returning its index if doesn’t exist, returning -1 |
| public static int methodD(int[] nums, int n){
  for(int i=0;i<nums.length;i++){
    if(i==n)
      return nums[i];
  }
  return -1;
} | returning the array element in the given index n if n out of bound, returning -1 |

(a) (8 points) Use your own words to explain what each method does (Fill in the table)

(b) (2 points) Which methods accomplish the same thing?
   methodA and methodD accomplish the same thing
   methodB and methodC accomplish the same thing
5. (7 points) Write a function `find36`. The function takes an array of integers as a parameter and returns a boolean value. It returns true if the array contains 3s or it contains 6s. For instance:

- `find36({1, 3, 5})` returns true
- `find36({2, 4, 6, 9})` returns true
- `find36({2, 5})` returns false

Method 1: Using for loop

```java
public static boolean find36(int[] nums) {
    for(int i = 0; i < nums.length; i++)
        if(nums[i]==3 || nums[i]==6)
            return true;
    return false;
}
```

Method 2: Using while loop

```java
public static boolean find36(int[] nums) {
    int i = 0;
    while(i < nums.length)
        if(nums[i]!=3 && nums[i]!=6)
            i++;
        else
            return true;
    return false;
}
```
6. (10 points) Write a function proArrays. The function takes two arrays of integer as parameters. It should return a new array of integers made of the product of the corresponding elements in the two arrays. Input arrays can have different lengths which are always greater or equal to 1. Any leftover elements go at the end of the result array.

proArrays({1,1,1},{2,2,2}) returns {2,2,2}
proArrays({1,2,3},{4,4,4,5,6}) returns {4,8,12,5,6}
proArrays({1,1,1,1},{2,3}) returns {2,3,1,1,1}

Method 1

public static int[] proArrays(int[] nums1, int[] nums2){
    int[] shortArray;
    int[] longArray;
    if (nums1.length < nums2.length) {
        shortArray = nums1;
        longArray = nums2;
    } else {
        shortArray = nums2;
        longArray = nums1;
    }
    int[] result = new int[longArray.length];
    for (int i = 0; i < shortArray.length; i++) {
        result[i] = shortArray[i] * longArray[i];
    }
    for (int i = shortArray.length; i < longArray.length; i++) {
        result[i] = longArray[i];
    }
    return result;
}
Method 2

```java
public static int[] proArrays(int[] nums1, int[] nums2) {
    int minLen = 0, maxLen = 0;
    if (nums1.length < nums2.length) {
        minLen = nums1.length;
        maxLen = nums2.length;
    } else {
        minLen = nums2.length;
        maxLen = nums1.length;
    }
    int[] result = new int[maxLen];
    int i = 0;
    for (i = 0; i < minLen; i++) {
        result[i] = nums1[i] * nums2[i];
    }
    if (maxLen != minLen) {
        if (nums1.length > nums2.length) {
            while (i < nums1.length) {
                result[i] = nums1[i];
                i++;
            }
        } else {
            while (i < nums2.length) {
                result[i] = nums2[i];
                i++;
            }
        }
    }
    return result;
}
```