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 5 questions on 7 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>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>50</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Consider the function below which is part of a Java program. Assume that sNums is always a sorted array.

```java
public static int what(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 `what`? `int`

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

(c) (1 point) How many parameter variables does the function `what` 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. (10 points) Write the output that the following code will print. You may assume the code compiles as written.

```java
public class ExamQuestion{
    public static int k = 1;
    public static int[] subtractK(int[] nums, int k){
        System.out.println(k);
        int[] result = nums;
        for(int i=0;i<nums.length;i++)
            result[i] = nums[i] - k;
        return result;
    }
    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);
        int[] arr2 = subtractK(arr1, ExamQuestion.k);
        System.out.println(Arrays.toString(arr2));
        System.out.println(Arrays.toString(arr2));
    }
}
```

1
[1, 2, 3, 4, 5]
2
[3, 4, 5, 6, 7]
2
1
[2, 3, 4, 5, 6]
[2, 3, 4, 5, 6]
3. (6 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><strong>public static int foo(char c) {</strong>&lt;br&gt;    return c + 1;&lt;br&gt; <strong>}</strong></td>
<td>No</td>
<td>c + 1 generates an int value</td>
</tr>
<tr>
<td><strong>String[] s = {&quot;a&quot;,&quot;b&quot;,&quot;c&quot;};</strong>&lt;br&gt;    <strong>System.out.println(s[1].charAt(1));</strong></td>
<td>Yes</td>
<td>s[1] is the string b and its second char doesn’t exist</td>
</tr>
<tr>
<td><strong>String s = &quot;book&quot;;</strong>&lt;br&gt; <strong>char[] c = s.toCharArray();</strong>&lt;br&gt; <strong>System.out.println(c[s.length()]);</strong></td>
<td>Yes</td>
<td>c is an array of size s.length so, we can’t access an element in the position equal to the length of the array</td>
</tr>
<tr>
<td><strong>int[] a; a[0] = 10;</strong></td>
<td>Yes</td>
<td>a is not initialized with a size</td>
</tr>
<tr>
<td><strong>public static boolean foo(int a) {</strong>&lt;br&gt;    <strong>if (a &gt; 10) return true;</strong>&lt;br&gt; <strong>}</strong></td>
<td>Yes</td>
<td>The method should return in all cases but it doesn’t return anything for a smaller or equal to 10</td>
</tr>
</tbody>
</table>
4. Consider the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>What?</th>
</tr>
</thead>
</table>
| public static int method1(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 the index if doesn’t exist, returning -1 |
| public static int method2(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 |
| public static int method3(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 method4(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 the index if doesn’t exist, returning -1 |

(a) (8 points) What is each method accomplishing? (Fill in the table)

(b) (2 points) Which methods are doing the same thing?
  
  method 1 and 4 do the same thing
  method 2 and 3 do the same thing


5. (14 points) Write a function `mixArrays`. The function takes two arrays of integer as parameters. It should return an array of integers made of the first element of the first array and the first element of the second array, followed by the second element of the first array and the second element of the second array, and so on. Input arrays can have different lengths. Any leftover elements go at the end of the result array.

`mixArrays({1,1,1},{2,2,2})` returns `{1,2,1,2,1,2}`
`mixArrays({1,2,3},{4,4,4,5,6})` returns `{1,4,2,4,3,4,5,6}`
`mixArrays({1,1,1,1,1},{2,3})` returns `{1,2,1,3,1,1,1,1,1}`

**Method 1: Using two indexes for given arrays**

```java
public static int[] mixArrays(int[] nums1, int[] nums2) {
    int[] result = new int[nums1.length + nums2.length];
    int nums1Index = 0;
    int nums2Index = 0;
    int i = 0;
    while(i<result.length){
        if(nums1Index < nums1.length)
            result[i++] = nums1[nums1Index++];
        if(nums2Index < nums2.length)
            result[i++] = nums2[nums2Index++];
    }
    return result;
}
```

**Method 2: Filling the elements for the shared indexes, then filling the left overs**

```java
public static int[] mixArrays(int[] nums1, int[] nums2) {
    int[] result = new int[nums1.length + nums2.length];
    int j = 0;
    int i = 0;
    for(i = 0; i < Math.min(nums1.length,nums2.length); i++){
        result[j++] = nums1[i];
        result[j++] = nums2[i];
    }
    while(i< nums1.length){
        result[j++] = nums1[i++];
    }
    while(i< nums2.length){
        result[j++] = nums2[i++];
    }
    return result;
}
```


    }
    
    return result;
    }
