Homework 4

Due: March 12, 11:59 PM

Daniel Carpenter

# Name:

113009743

**Student ID:**

**Submission Instructions:** This PDF contains fillable fields where you can input your answers. (For example, you can input your name and ID in the fields above.) Please save the document regularly so your answers are not lost. After you complete the assignment, upload a copy to Gradescope. A link to Gradescope can be found on the left side of the course Canvas page.

**Important Note on Academic Integrity:** This assignment should be completed individually. In recent semesters, improper collaboration on homework has led to multiple cases of plagiarism, where we receive identical or nearly identical submissions from two or more students. If you decide to discuss this assignment with other students before the deadline, make sure you first read the section of the syllabus on proper and improper collaboration. Additionally, you must include the names of these students below.

**Names of Collaborators (if any):**

**Question 1 (10 points; 2 per part):** Suppose that an array is declared and constructed with the following line of code:

int[] lostNums = {-7, 4, 9, -5, 11, 14};

Answer each of the questions below about the array.

1. What is the index of the element 9?

2

1. What is the value of the field lostNums.length?

6

1. What is the value of the element lostNums[1]?

4

1. Suppose that we sort the array using the sort method from the Arrays class:

Arrays.sort(lostNums);

Now we can use binary search to find the location (i.e., index) of any element in the array. What value is returned by the method call below?

Arrays.binarySearch(lostNums, 11);

4

1. We can also use binary search to look for an element that is not in the array (that you may assume is still sorted). Read the API documentation for the binarySearch method carefully to answer this question. What value is returned by the method call below?

Arrays.binarySearch(lostNums, 2);

-3

**Question 2 (30 points; 6 per part):** Trace the execution of each program in the memory diagrams. Variables declared in the header of a for-loop do not need to be traced, but every other variable should appear in a table. Be sure to include the initial value of each variable. If the contents of a variable change, use a comma, space, or some other delimiter to separate the old and new values.



public class Example

{

public static void main(String[] args)  
{  
 int[] notPrimes = {1, 4, 6};  
 int factor = 2;   
 notPrimes = scale(notPrimes, factor);  
}  
  
public static int[] scale(int[] arr, int mult)  
{  
 int[] product = new int[arr.length\*2];  
  
 for (int idx = 0; idx < arr.length; ++idx)  
 {  
 product[idx\*2] = mult \* arr[idx];

product[idx\*2+1] = mult – arr[idx];  
 }  
  
 return product;  
}

}

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| notPrimes | 100 | 1000, 1004 |
| factor | 101 | 2 |
|  | 102 |  |
|  | 103 |  |

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| 0 | 1000 | 1 |
| 1 | 1001 | 4 |
| 2 | 1002 | 6 |
| length | 1003 | 3 |
| 0 | 1004 | 0, 2 |
| 1 | 1005 | 0, 1 |
| 2 | 1006 | 0, 8 |
| 3 | 1007 | 0, -2 |
| 4 | 1008 | 0, 12 |
| 5 | 1009 | 0, -4 |
| length | 1010 | 6 |
|  | 1011 |  |

**main Stack Frame Heap**

**scale Stack Frame**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arr | 200 | 1000 |
| mult | 201 | 2 |
| product | 202 | 1004 |
| idx | 203 | 0, 1, 2 |

public class Example

{

public static void main(String[] args)  
{  
 int[] notPrimes = {1, 4, 6};  
 int factor = 3;  
 scale(notPrimes, factor);  
}  
  
public static void scale(int[] arr, int mult)  
{  
 for (int idx = 0; idx < arr.length; ++idx)  
 {  
 arr[idx] = mult - arr[idx];  
 }  
}

}

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| notPrimes | 100 | 1000 |
| factor | 101 | 3 |
|  | 102 |  |
|  | 103 |  |

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| 0 | 1000 | 1, 2 |
| 1 | 1001 | 4, -1 |
| 2 | 1002 | 6, -3 |
| length | 1003 | 3 |
|  | 1004 |  |
|  | 1005 |  |
|  | 1006 |  |
|  | 1007 |  |
|  | 1008 |  |
|  | 1009 |  |
|  | 1010 |  |
|  | 1011 |  |

**main Stack Frame Heap**

**scale Stack Frame**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arr | 200 | 1000 |
| mult | 201 | 3 |
| idx | 202 | 0, 1, 2 |
|  | 203 |  |



public class Example

{

public static void main(String[] args)  
{  
 int[] arrayIn = {9, 3, 7, 4};  
 repeatArray(arrayIn);  
}  
  
public static int[] repeatArray(int[] arr)  
{  
 int[] repeated = new int[2\*arr.length];  
  
 for (int idx = 0; idx < arr.length; ++idx)  
 {  
 int element = arr[idx];  
 repeated[idx] = element;  
 repeated[idx + arr.length] = element;  
 }  
  
 return repeated;  
}

}

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arrayIn | 100 | 1000, 1005 |
|  | 101 |  |
|  | 102 |  |
|  | 103 |  |

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| 0 | 1000 | 9 |
| 1 | 1001 | 3 |
| 2 | 1002 | 7 |
| 3 | 1003 | 4 |
| length | 1004 | 4 |
| 0 | 1005 | 0, 9 |
| 1 | 1006 | 0, 3 |
| 2 | 1007 | 0, 7 |
| 3 | 1008 | 0, 4 |
| 4 | 1009 | 0, 9 |
| 5 | 1010 | 0, 3 |
| 6 | 1011 | 0, 7 |
| 7 | 1012 | 0, 4 |
| length | 1013 | 8 |
|  | 1014 |  |
|  | 1015 |  |
|  | 1016 |  |

**main Stack Frame Heap**

**repeatArray Stack Frame**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arr | 200 | 1000 |
| repeated | 201 | 1005 |
| idx | 202 | 0 |
| element | 203 | 9, 3, 7, 4 |

public class Example

{

public static void main(String[] args)  
{  
 int[] arrayIn = {7, 8, 9, 10, 12, 14};  
 int shift = 2;  
 shiftRight(arrayIn, shift);  
}  
  
public static void shiftRight(int[] arr, int offset)  
{  
 int[] temp = new int[arr.length];  
  
 for (int idx = 0; idx < arr.length; ++idx)  
 {  
 int idxOffset = (idx + offset) % arr.length;  
 temp[idxOffset] = arr[idx];  
 }  
  
 for (int idx = 0; idx < arr.length; ++idx)  
 {  
 arr[idx] = temp[idx];  
 }  
}

}

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arrayIn | 100 | 1000 |
| shift | 101 | 2 |
|  | 102 |  |
|  | 103 |  |

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| 0 | 1000 | 7, 12 |
| 1 | 1001 | 8, 14 |
| 2 | 1002 | 9, 7 |
| 3 | 1003 | 10, 8 |
| 4 | 1004 | 12, 9 |
| 5 | 1005 | 14, 10 |
| length | 1006 | 6 |
| 0 | 1007 | 0, 12 |
| 1 | 1008 | 0, 14 |
| 2 | 1009 | 0, 7 |
| 3 | 1010 | 0, 8 |
| 4 | 1011 | 0, 9 |
| 5 | 1012 | 0, 10 |
| length | 1013 | 6 |

**main Stack Frame Heap**

**shiftRight Stack Frame**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arr | 200 | 1000 |
| offset | 201 | 2 |
| temp | 202 | 1007 |
| idx | 203 | 0,1,2,3,4,5 |
| idxOffset | 204 | 2,3,4,5,0,1 |



public class Example

{

public static void main(String[] args)  
{  
 int[] arr1 = {7, 1, 6, 5, 2};  
 int[] arr2 = {3, 9, 4};  
 int[] sum = addArrays(arr2, arr1); // Careful!  
}  
  
public static int[] addArrays(int[] arr1, int[] arr2)  
{  
 if (arr1.length < arr2.length)   
 {  
 int[] temp = arr1;  
 arr1 = arr2;  
 arr2 = temp;  
 }  
   
 int[] sum = new int[arr1.length];  
 for (int idx = 0; idx < arr1.length; ++idx)   
 {  
 sum[idx] = arr1[idx];  
 }  
  
 for (int idx = 0; idx < arr2.length; ++idx)  
 {  
 sum[idx] = arr1[idx] + arr2[idx];  
 }  
  
 return sum;  
}

}

// Diagrams are on the next page

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arr1 | 100 | 1000 |
| arr2 | 101 | 1006 |
| sum | 102 | 1010 |
|  | 103 |  |

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| 0 | 1000 | 7 |
| 1 | 1001 | 1 |
| 2 | 1002 | 6 |
| 3 | 1003 | 5 |
| 4 | 1004 | 2 |
| length | 1005 | 5 |
| 0 | 1006 | 3 |
| 1 | 1007 | 9 |
| 2 | 1008 | 4 |
| length | 1009 | 3 |
| 0 | 1010 | 10 |
| 1 | 1011 | 10 |
| 2 | 1012 | 10 |
| 3 | 1013 | 5 |
| 4 | 1014 | 2 |
| length | 1015 | 5 |
|  | 1016 |  |
|  | 1017 |  |
|  | 1018 |  |
|  | 1019 |  |

**main Stack Frame Heap**

**addArrays Stack Frame**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Address** | **Contents** |
| arr1 | 200 | 1000, 1006 |
| arr2 | 201 | 1006, 1000 |
| temp | 202 | 1000 |
| sum | 203 | 1010 |

**Question 3 (10 points; 2 per part):** Write the signature for a method that performs each of the computations described below. **You do not need to write the body of the method.**

Example: Return a copy of an array of boolean values.

boolean[] copy(boolean[] array)

1. Subtract a given value from all of the elements in an array of ints.

Int[] subractFromArray(int value, int[] array)

1. Subtract a given value from all of the elements in a copy an array of ints and return the copy.

Int[] subractFromArray(int value, int[] array)

1. Find whether or not a given character appears in a String.

Boolean charInString(char varChar, String varString)

1. Find the sum of the elements in an array of integers with an index in a given range.

Int[] sumOfRange(int lowerBound, intUpperBound, int[] array)

1. Return an array of a given length where all the elements are equal to a given int.

Int[] scalerArray(int length, int scaler, int[] array)