**CS 515 Exercise A06.1: Merge-Sort**

\_\_\_\_\_ /50

**Names: Liam Warren Wildcats ID’s : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lecture Section: 01 / 02 (circle one)**

**[20 pts.]** How would you write a method to merge two arrays into a new array?

import java.util.Arrays;

public class Merge

{

public static String[] merge(String[] arrA, String[] arrB)

{

String[] arrC = new String[arrA.length + arrB.length];

Int aIndex = 0;

Int bIndex = 0;

// Since the size of arrC is defined as the sum of arrA and // arrB’s sizes we know we have exactly enough items to // fill arrC

For(int I =0, I < arrC.length, ++i) {

If (arrA.length < index) {

//if we run out of items in arrA

arrC[i] = arrB[index];

++bIndex;

}else if (arrB.length - 1 < index) {

// If we run out of items in arrB

arrC[i] = arrA[index];

++aIndex;

}else If (arrA[index] <= arrB[index]){

// check index of arrA and arrB, put the smaller

//in arrC

arrC[i] = arrA[index];

++aIndex;

} else {

arrC[i] = arrB[index];

++bIndex

}

Return arrC;

}

}

*More on the back…*

**[30 pts.]** Sort the array below using Merge-Sort. Assume that the array has already been broken into pieces by repeated halving. Then use the first empty row to properly order the initial arrays of sizes 1 and 2. In the empty rows that follow, show how each adjacent pair of arrays would be merged at each step.

Initial array:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 9 | 1 | 3 | 5 | 8 | 11 | 15 | 7 | 4 | 14 | 2 | 13 | 12 | 6 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **1** | **9** | **3** | **5** | **8** | **11** | **7** | **15** | **4** | **14** | **2** | **13** | **6** | **12** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **9** | **10** | **3** | **5** | **8** | **11** | **4** | **7** | **14** | **15** | **2** | **6** | **12** | **13** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **3** | **5** | **8** | **9** | **10** | **11** | **2** | **4** | **6** | **7** | **12** | **13** | **14** | **15** |

Final array:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |