**Homework Assignment 4**

Due date: October 6th, 11:55pm EST

**Problem 1**

Quick sort

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 12 | 13 | 27 | 65 | **43** | 59 | 45 | 67 | 7 | 5 | 55 |

Pivot = 4th index (43) -> moved to the end

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | **10** |
| value | 12 | 13 | 27 | 65 | 55 | 59 | 45 | 67 | 7 | 5 | **43** |

Swap 3 and9, since 65> 43, and 5 < 43

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | **10** |
| value | 12 | 13 | 27 | 5 | 55 | 59 | 45 | 67 | 7 | 65 | **43** |

Swap 4 and 8, since 55 > 43 and 7 < 43

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | **10** |
| value | 12 | 13 | 27 | 5 | 7 | 59 | 45 | 67 | 55 | 65 | **43** |

Bounds cross at 5, so switch with the pivot

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 12 | 13 | 27 | 5 | 7 | 43 | 45 | 67 | 55 | 65 | 59 |

Makes a new pivot, and moves it to the last, switching 2 and 4

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | **4** | *5* | 6 | 7 | 8 | 9 | 10 |
| value | 12 | 13 | 7 | 5 | **27** | *43* | 45 | 67 | 55 | 65 | 59 |

Bounds have crossed, picks a new pivot, switching 1 and 3

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | **3** | 4 | *5* | 6 | 7 | 8 | 9 | 10 |
| value | 12 | 5 | 7 | **13** | 27 | *43* | 45 | 67 | 55 | 65 | 59 |

Bounds have crossed, picks a new pivot, switching 1 and 2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | **2** | 3 | 4 | *5* | 6 | 7 | 8 | 9 | 10 |
| value | 12 | 7 | **5** | 13 | 27 | *43* | 45 | 67 | 55 | 65 | 59 |

Bounds have crossed at 0, swap the pivot (2) with 0. Left side of the array is sorted

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | *5* | 6 | 7 | 8 | 9 | 10 |
| value | 5 | 7 | 12 | 13 | 27 | *43* | 45 | 67 | 55 | 65 | 59 |

Picks a pivot (8) and moves it to the end

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | *5* | 6 | 7 | 8 | 9 | **10** |
| value | 5 | 7 | 12 | 13 | 27 | *43* | 45 | 67 | 59 | 65 | **55** |

Bounds cross at 7, swaps 7 with the pivot- 6 is alone and already sorted.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | *5* | 6 | 7 | 8 | 9 | 10 |
| value | 5 | 7 | 12 | 13 | 27 | *43* | 45 | 55 | 59 | 65 | 67 |

Picks a pivot (9) and moves it to the end

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | *5* | 6 | 7 | 8 | 9 | **10** |
| value | 5 | 7 | 12 | 13 | 27 | *43* | 45 | 55 | 59 | 67 | **65** |

Bounds cross at 9, swaps 9 with the pivot, right side of the array is sorted.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | *5* | 6 | 7 | 8 | 9 | 10 |
| value | 5 | 7 | 12 | 13 | 27 | *43* | 45 | 55 | 59 | 65 | 67 |

**Problem 2**

Merge Sort – splits the array in half

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 76 | 27 | 62 | 43 | 59 | 45 | 87 | 13 | 5 | 99 |

Splits the array in 2 again

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 76 | 27 | 62 | 43 | 59 | 45 | 87 | 13 | 5 | 99 |

0 and 1 values are sorted in that batch

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 76 | 27 | 62 | 43 | 59 | 45 | 87 | 13 | 5 | 99 |

0, 1 and 2 are sorted in that batch

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 27 | 76 | 62 | 43 | 59 | 45 | 87 | 13 | 5 | 99 |

3, 4 and 5 are sorted in that batch

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 27 | 76 | 43 | 59 | 62 | 45 | 87 | 13 | 5 | 99 |

First half is sorted

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 27 | 43 | 59 | 62 | 76 | 45 | 87 | 13 | 5 | 99 |

First section of the second half is sorted

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 27 | 43 | 59 | 62 | 76 | 13 | 45 | 87 | 5 | 99 |

Second section of the second half is sorted

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 27 | 43 | 59 | 62 | 76 | 13 | 45 | 87 | 5 | 99 |

Second half is sorted

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 27 | 43 | 59 | 62 | 76 | 5 | 13 | 45 | 87 | 99 |

Entire array is sorted

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| value | 2 | 5 | 13 | 27 | 43 | 45 | 59 | 62 | 76 | 87 | 99 |

**Problem 3**

|  |
| --- |
| **public** **static** <E **extends** Comparable<E>> E[] mergeArrays(E[] array1, E[] array2)  {  // Instantiation of pointers  **int** pointer1 = 0;  **int** pointer2 = 0;  // Instantiation of the final large merged array  **int** mergedArraySize = array1.length + array2.length;    @SuppressWarnings("unchecked") // Gets rid of the warnings for casting a generic object  E[] mergedArrays = (E[]) Array.*newInstance*(array1.getClass().getComponentType(), mergedArraySize);    // Generates a for loop that will place the lower of the two values  **for** (**int** i = 0; i < mergedArrays.length; ++i)  {  // Stops the loop if it reaches the last value that has nothing to compare it to  **if** (pointer1 == array1.length)  {mergedArrays[i] = array2[pointer2];}  **if** (pointer2 == array2.length)  {mergedArrays[i] = array1[pointer1];}    // If the value from array2 is lower:  **else** **if** (array2[pointer2].compareTo(array1[pointer1]) < 0)  {  mergedArrays[i] = array2[pointer2]; // Adds the value to the mergedArray  ++pointer2; // Increments the second array's pointer  }  // If the value from array1 is lower, or if they are equal:  **else**  {  mergedArrays[i] = array1[pointer1]; // Adds the value to the mergedArray  ++pointer1; // Increments the first array's pointer  }  } // End of merging the two arrays  **return** mergedArrays;  } // End of mergerArrays method |