**Merge Sort Homework**

Use the program we went over in class: Prog260MergeSort.zip

Put in a break point at 2 locations, **here**: (the code has comments pointing these locations out)

public static void MergeSort\_Recursive(int[] inputArray, int sLeftPointer, int sRightPointer)

{ **// add breakpoint here**

**And then at the end of**

private static void DoMerge(int[] inputArray, int mLeftPointer, int mMidPointer, int mRightPointer)

.

.

.

for (i = 0; i < num\_elements; i++) // this copies over the values back to the orig array

{

inputArray[mRightPointer] = temp[mRightPointer];

mRightPointer--;

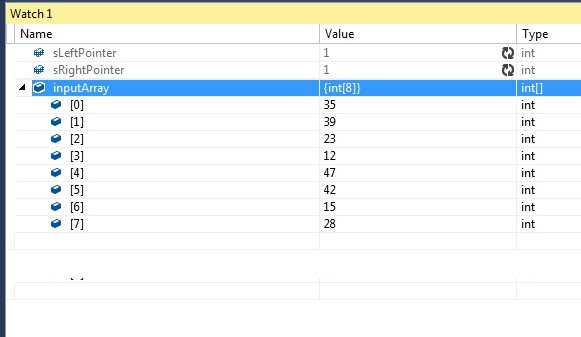
}

} **// second break point here**

In the program, make sure it is set to use the small array, not the big one ( int[] theArray = testArray1; )

Run the program, it should stop at the first breakpoint you added.

Then add watches for all these items: sLeftPointer sRightPointer, and inputArray



Now gather data from your program and document the sort process in a WORD, or PPT document. The big idea is to:

* show the sequence of calls to MergeSort\_Recusive and DoMerge,
* document and understand what segment of the array is currently being worked on,
* and what the state of the array looks like at that time.

For **each** call into

public static void MergeSort\_Recursive(int[] inputArray, int sLeftPointer, int sRightPointer)

document the state of inputArray, sLeftPointer, and sRightPointer. Not only do you need to show the values, but you must document you understand where in the sort process the code is. Do this by showing the state of the sort like this or some other reasonable way…

Example --- NOT ACCURATE

Entering MergeSort\_Recursive the 1st time, the pointers are, and the array is

MS1 0 - 7 [39 35 23 12 47 42 15 28 ]

^ method (MS) and which time 1, or 1st

^ left and right pointers 0 and 7

Therefore, the segment of the array being processed at this moment (all 8 elements)

The second time:

Entering MergeSort\_Recursive 2nd time, the pointers are, and the array is

MS2 0 - 3 [39 35 23 12] [ 47 42 15 28 ]

^ method (MS) and which time 2, or 2nd

^ left and right pointers 0 and 3

Therefore, the segment of the array being processed at this moment (just the first four elements)

Entering MergeSort\_Recursive 3rd time, the pointers are, and the array is

MS3 0 - 1 [39 35] [23 12] [ 47 42 15 28 ]

What I am showing is how the pointers are breaking the array down into sub arrays, with the red text showing the current subset of the array that is currently being processed.

Eventually you will see calls to **DoMerge**, at which point the numbers in the arrays start to swap locations.

So your task is to document 2 things clearly

1. How the code breaks the problem down into smaller pieces, and as you enter MergeSort\_Recursive each time, **what part of the array is it working on**. **Place [ ] brackets** as I have shown to indicate the state of the sort and **use colors** to show the subset that is current.
2. How the DoMerge starts to flip elements to get the array in order, and again, use colors to show what part was just changed before leaving DoMerge.

1st - About to leave DoMerge, having just merged the array of 1 element, 35 and the array of 1 element, 39

DM1 [35 39] [23 12] [47 42 15 28]

Then after few more calls to MergeSort\_Recursive …

Another calls to DoMerge

DM2 [35 39] [12 23] [47 42 15 28]

This is the 2nd call to Do merge, and it just merged the 12 and 23 to be in order.

Etc.

Eventually you will get to the last call to DoMerge, ending with (having just merged the left array of 4 items and the right array of 4 items into the final sort.

DM7 [12 15 23 28 35 39 42 47]

So you document will show, in one sequence, every call to MergeSort\_Recursive and DoMerge and what part of the array is being worked on.