Objective: Become more proficient at implementing sorting algorithms.

Part A: Implement a modified bubble sort algorithm to alternate “bubbling” in both directions. The first pass through the “unsorted” part of the array bubbles the largest item to the right end of the “unsorted” part. The second pass through the “unsorted” part of the array bubbles the smallest item to the left end of the “unsorted” part. This alternating pattern continues until no more passes are necessary.

<table>
<thead>
<tr>
<th>sorted part</th>
<th>unsorted items</th>
<th>sorted part</th>
</tr>
</thead>
<tbody>
<tr>
<td>(smallest items)</td>
<td></td>
<td>(largest items)</td>
</tr>
</tbody>
</table>

Include a timing program as in lab8.zip for your program that times your sorting algorithm several times with different initial orderings of 15,000 list items. The initial orderings of items are: descending order, ascending order, random order, and random order again to check for consistency. Report the times and compare your times to the original bubble sort from lab8. Include an explanation of the timing comparisons.

Part B: Combining Quick Sort and Merge Sort

Quicksort has both advantages and disadvantages:
- Advantage: only a single list is needed with items just being rearranged within it (i.e., no extra storage needed)
- Disadvantage: on “average” quick sort is $O(n \log_2 n)$. However, if the pivot item repeated falls at the end of the section to sort, then quick sort can be $O(n^2)$ in the worst case.

Merge sort also has both advantages and disadvantages:
- Advantage: even in the worst case, merge sort is $O(n \log_2 n)$ because we always break the lists in half.
- Disadvantage: extra storage is needed when splitting the lists into two smaller halves.

I’d like you to combine quick sort and merge sort to mostly keep the advantages of both. Suppose we divide the original list to sort into roughly 16 equal pieces and perform quicksort on each piece. Now, we can merge sorted pairs of pieces back together as the following diagrams shows using only one additional list of size n (i.e., the “temp” list)
Include a timing program as in lab8.zip for your combined quick and merge sort program that times the sorting of randomly generated lists of sizes 1,000,000 items and 2,000,000 items. Report these times and compare your times to the original 2-way merge sort and original quick sort from lab 8. Include an explanation of the timing comparisons.

SUBMISSION
Submit **ALL necessary files** to run your sorts and your timing “reports” for parts A and B as a single zipped file (called hw5.zip) electronically at:

https://www.cs.uni.edu/~schafer/submit/which_course.cgi