This exercise augments classroom discussion of the AVR application note on bubble-sort[1] and its accompanying code `avr220.asm`.

1. List the members of your group below. Underline your name.

2. What is the size of a *word* as used in the note to measure code size?

3. What is the maximum amount of data that the routine can sort? What is the source of this restriction? How may it be removed?

4. Provide an alternate description of the bubble-sort algorithm outlined by steps 1 through 8 on the first page of the note. Aim for a description that is short, precise, and assumes no knowledge beyond that assumed by the note's description.

5. Comment on the note's characterization of the algorithm's running time near the bottom of the first page.

[1] Atmel Corporation, AVR220: Bubble Sort, `http://www.atmel.com/`, 2002.

6. Comment on the pseudocode on page 2 of the note. Provide alternate pseudocode and highlight the differences.

7. Briefly explain the notation `endH:endL` as used on page 2 of the note.

8. Explain, as precisely as possible, how the algorithm description on page 2 follows from the earlier descriptions of bubble-sort.

9. Provide a precise English description of the *Shellsort* algorithm. Indicate which *increment sequence* is used.

10. Provide C, Java, or Python code for the algorithm of Question 9. *Briefly explain why your code is correct.*

11. Provide a concise implementation of Shellsort in AVR assembler, using the bubble-sort implementation as a guide. [Hint: It may be useful to reuse most of the bubble-sort code.]