Answers to Bubble Sort Exercises

1. (a) 20 comparison, 12 swaps

   (b) Largest element (8) has index 6; smallest element (2) has index 0

   (c) Element X[3] has the value 5

2. (a) 20 comparison, 11 swaps

   (b) Largest element (7) has index 0; smallest element (-4) has index 6

   (c) Element X[4] has the value -1

3. (a) Since the array is initially in descending order and a descending sort has been requested, one pass of 6 comparisons will result in 0 swaps, causing the sort to end

   (b) Actual results agree with the predictions.

   (c) Since the array is initially sorted in descending order but an ascending sort has been requested, there will be six passes required, with 6+5+4+3+2+1 = 21 comparisons. Each comparison will require a swap, also resulting in a total of 21 swaps.

   (d) Actual results agree with the predictions.

4. (a) 20 comparisons, 9 swaps
   (b) If the sort is ascending, the OzoBlockly bubble sort subroutine only swaps if left > right, not if they are equal. If the sort is descending, the OzoBlockly bubble sort subroutine only swaps if left < right, not if they are equal. This seems quite reasonable, as there is no need to switch equal values. Yes.
5. (a) In a best case scenario (items initially sorted in the correct sort order) with an array of 100 integers, just one pass of 99 comparisons and 0 swaps would be required.
   (b) In a worst case scenario (items initially sorted in reverse of the requested sort order) with an array of 100 integers, 99+98+97+ … + 3+2+1 = 99*100/2 = 4950 comparisons would be required, and with each comparison, a swap would also be required
   (c) 999 comparisons and 0 swaps would be required, by the reasoning of part (a).
   (d) 999*1000/2 = 499,500 swaps and 499,500 comparisons would be required. Wow! This is explained by the reasoning of part (b).
   (e) (i) As the number of input values (n) tends toward infinity, the number of comparisons (n-1) also tends toward infinity. O(n) best describes the time complexity in this case.
   (e) (ii) As the number of input values (n) tends toward infinity, the number of comparisons (and the number of swaps) tend toward (n-1)*n/2. $O(n^2)$ best describes the time complexity in this case.
6. (a) Simply change the array declaration block to a length of 5. (Note that you would also have to remove the intersections in the Ozomap with indexes 5 and 6.)
   (b) The variable *temp* holds the value of *arg1*, so that it can be assigned to *X(i + 1)* in the second "set element" block in the Swap subroutine.