A. (a) False.

$f(n) + g(n) = \Omega\Big(\min(f(n), g(n))\Big)$, since $f(n) + g(n) \geq \min(f(n), g(n))$.

However, if we take $f(n) = n$ and $g(n) = n^2$, then $f(n) + g(n) = n + n^2$ and $\min\Big(f(n), g(n)\Big) = n$. But there is no positive constant c such that $n + n^2 \leq cn$ for arbitrarily large n. (Given c, $n + n^2 \leq cn$ implies that $n \leq c - 1$.) Hence $f(n) + g(n) \neq O\Big(\min(f(n), g(n))\Big)$, so $f(n) + g(n) \neq \Theta\Big(\min(f(n), g(n))\Big)$.

(b) True.

We need to find positive constants c_1, c_2 and n_0 such that

$$c_1\Big(\max(f(n), g(n))\Big) \;\leq\; f(n) + g(n) \;\leq\; c_2\Big(\max(f(n), g(n))\Big)$$

for all $n \geq n_0$.

The first inequality clearly holds with $c_1 = 1$ (and $n_0 = 1$), and the second inequality clearly holds with $c_2 = 2$ (and $n_0 = 1$).

Hence the constants we seek are $c_1 = 1$, $c_2 = 2$, and $n_0 = 1$.

B. (a)

$$T(n) = 1 + T(n-2) = 2 + T(n-4) = 3 + T(n-6) = \cdots$$
$$= \lfloor n/2 \rfloor + T(n - 2\lfloor n/2 \rfloor) = \begin{cases} \lfloor n/2 \rfloor + T(0), & \text{if } n \text{ is even;} \\ \lfloor n/2 \rfloor + T(1), & \text{if } n \text{ is odd.} \end{cases}$$

Hence $\underline{T(n) = \Theta(n)}$.

(b) By the Master Theorem with $a = 9$, $b = 4$ and $c = 2$ (case 3 with $c = 2 > \log_4 9 = \log_b a$), $\underline{T(n) = \Theta(n^2)}$.

(c) By the Master Theorem with $a = 3$, $b = 2$ and $c = 1$ (case 1 with $c = 1 < \log_2 3 = \log_b a$), $\underline{T(n) = \Theta(n^{\log_2 3})}$.

C. (a) We can prove that $\textsc{TriSort}(A, i, j)$ correctly sorts the subarray $A[i..j]$ by induction on $j{-}i$.

If $j{-}i<1$ then there is nothing to do, and the algorithm does nothing.

If $j{-}i = 1$, then the **if**-statement on line 1 ensures that the two values in the subarray are properly ordered.

For the induction step, assume that TriSort correctly sorts subarrays of length $< j{-}i$. The effect of the three recursive calls are as follows:

- After $\textsc{TriSort}(A, i, j{-}k)$, all elements in the first two-thirds of the subarray that belong in the last one-third of the subarray will have been moved to the middle one-third of the subarray.
- After $\textsc{TriSort}(A, i{+}k, j)$, all elements that belong in the last one-third of the subarray will have been moved to their correct positions in the last one-third of the subarray.
- After $\textsc{TriSort}(A, i, j{-}k)$, all remaining elements will have been moved to their correct positions.

(b) The time complexity of TriSort is captured by the recurrence

$$T(n) \;=\; 3T(2n/3) \;+\; \Theta(1).$$

By the Master Theorem (case 1, with $a = 3$, $b = 3/2$ and $c = 0$), the asymptotic worst-case running time of TriSort is

$$T(n) \;=\; \Theta\left(n^{\log_{3/2} 3}\right) \;\approx\; \Theta(n^{2.71}).$$

This is a particularly bad sorting algorithm, as it is asymptotically worse than the worst-case running time of $\Theta(n^2)$ for InsertionSort, and the worst-case running time of $\Theta(n \lg n)$ for MergeSort.