Question 1. $5, 8, 11$

Question 2. The first statement is true, the second is false. To see that this is so, imagine creating an undirected graph that has a vertex for each of the items, and adding an undirected edge betwen two vertices if an only if the algorithm compares them. The resulting graph cannot be connected unless it has $2n$ edges, and therefore such a graph is disconnected for any algorithm that does fewer than $2n$ comparisons: Let C be a connected component that does not contain the median (call it $\hat{x}$) that is returned by the algorithm. Let M be an integer larger than any of the $2n+1$ input numbers. Run the algorithm 2 more times, once with a $+M$ added to each item in component C, and another time with a $-M$ added to each item in component C: In both cases the outcomes of the comparisons made by the algorithm stay the same and therefore in both cases it returns the same old $\hat{x}$ as the median, which cannot be the correct answer for both of the 2 extra runs of the algorithm (at most one of them can result in the same $\hat{x}$ remaining the median).

Question 3. We prove the lower bound by giving a linear-time reduction from `set equality`. For any instance A, B of `set equality`, create the following instance of the problem at hand. First, make all the numbers in A and B positive, by adding the same large enough positive number to all of them; this can be done in $O(n)$ time, and does not change the answer to the set equality question. Assume that this has already been done, and that the resulting two sets of positive numbers are $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$. The instance of the "symmetry" problem that we create is

$$\{0, a_1, \ldots, a_n, -b_1, \ldots, -b_n\}$$

in which the median is 0, and the "symmetry" property holds if and only if the two sets are equal. Note that the above reduction would not work without first making all the numbers positive (for example, it fails when A and B are completely different but each of them has 0 as its own center of symmetry).

Question 4. It is possible to sort in $O(nT(n))$ as follows: Starting with an empty data structure, we insert the n numbers to be sorted, and then perform n consecutive ExtractMin operations which give the numbers in sorted order. This, together with the $\Omega(n \log n)$ time lower bound for sorting, imply that $T(n) = \Omega(\log n)$ (because otherwise we could sort faster than the lower bound for sorting, a contradiction).

Question 5. $(S2, S4)$, $(S2, S5)$, $(S4, S5)$, $(S3, S6)$, $(S6, S7)$, $(S3, S7)$