1.
```
ThreeSmallest(A, n):
    if n = 3:    return solve_three(A[0], A[1], A[2])
    if n = 4:    return solve_four(A[0], A[1], A[2], A[3])
    if n = 5:    return solve_five(A[0], A[1], A[2], A[3], A[4])
    m = ⌊n/2⌋
    (a, b, c) = ThreeSmallest(A[0 ... m − 1])
    (d, e, f) = ThreeSmallest(A[m ... n − 1])
    # Perform a "partial merge" of (a, b, c) and (d, e, f) to obtain the three smallest elements.
    if a < d:
        if b < d:
            if c < d:    return (a, b, c)
            else:        return (a, b, d)
        else:  # d < b
            if b < e:    return (a, d, b)
            else:        return (a, d, e)
    else:  # d < a
        if a < e:
            if b < e:    return (d, a, b)
            else:        return (d, a, e)
        else:  # e < a
            if a < f:    return (d, e, a)
            else:        return (d, e, f)
```

This satisfies the recurrence for $C(n)$ exactly, since the first recursive call is made on an input of size $\lfloor n/2 \rfloor$, the second on an input of size $\lceil n/2 \rceil$, and the algorithm performs 3 more comparisons between list elements during its "partial merge" phase at the end.

2. The Master Theorem applies to the recurrence for $C(n)$, with $a = 2$ (two recursive calls are made), $b = 2$ (each recursive call is on an input roughly half the size), and $d = 0$ (the algorithm carries out a constant amount of work outside the recursive calls).

   Since $a = 2 > 1 = b^d$, the Master Theorem allows us to conclude that $C(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 2}) = \Theta(n)$.

3. CLAIM: $\forall n \geqslant 3, C(n) = 2n - 3$.

   PROOF: By complete induction on $n \geqslant 3$.

   Base Cases: We show that every initial value of $C(n)$ satisfies the statement:

$$
\begin{aligned}
C(3) &= 3 = 6 - 3 = 2(3) - 3 \\
C(4) &= 5 = 8 - 3 = 2(4) - 3 \\
C(5) &= 7 = 10 - 3 = 2(5) - 3
\end{aligned}
$$

   Ind. Hyp.: Assume $n \geqslant 6$ and $C(k) = 2k - 3$ for $k \in \{3, 4, \ldots, n-1\}$.

   Ind. Step:
$$
\begin{aligned}
C(n) &= C(\lceil n/2 \rceil) + C(\lfloor n/2 \rfloor) + 3 && \text{(since } n \geqslant 6\text{)} \\
&= (2\lceil n/2 \rceil - 3) + (2\lfloor n/2 \rfloor - 3) + 3 && \text{(by the I.H.)} \\
&= 2(\lceil n/2 \rceil + \lfloor n/2 \rfloor) - 3 - 3 + 3 \\
&= 2n - 3
\end{aligned}
$$

   Conclusion: By induction, $\forall n \geqslant 3, C(n) = 2n - 3$.

   This means our divide-and-conquer algorithm is better than the naive algorithm, performing $n - 3$ fewer comparisons on inputs of size n.