There are 2 questions for a total of 10 points.

---

1. Apply the Master theorem and give the solution for the following recurrence relations in big-O notation. Explanation is not required.
   (a) (1 point) $T(n) = 2 \cdot T(n/2) + O(1); T(1) = O(1)$

   Solution: $T(n) = O(n)$.

   (b) (1 point) $T(n) = 2 \cdot T(n/2) + O(n); T(1) = O(1)$

   Solution: $T(n) = O(n \log n)$.

   (c) (1 point) $T(n) = 2 \cdot T(n/2) + O(n^3); T(1) = O(1)$

   Solution: $T(n) = O(n^3)$.

2. (7 points) You are given a bit-array $A[1...n]$ (i.e., $A[i] \in \{0, 1\}$ for each i) and told that this is a "0-to-1" bit-array. This means that for some (unknown) index $1 \leq j < n$, $A[1], ..., A[j]$ are all 0's and $A[j + 1], ..., A[n]$ are all 1's. The index j for such an array is called the transition index.

   Design an algorithm for finding the transition index for a given 0-to-1 bit-array. The input to your algorithm is an array A and the size n of the array A. Give a running time analysis for your algorithm.

Solution: Here is a divide and conquer based algorithm for the problem that follows the binary search idea.

```
FindTransition(A, n)
  - return(RecFind(A, 1, n))

RecFind(A, i, j)
  - If (j = i + 1) return(i)
  - mid ← ⌊(i+j)/2⌋
  - If (A[mid] = 0) return(FindTransition(A, mid, j))
  - else return(FindTransition(A, i, mid))
```

Running time: The recurrence relation for the recursive program is $T(n) = T(n/2)+O(1)$. Applying the Master theorem with $a = 1; b = 2; d = 0$ (steady state), we obtain $T(n) = O(\log n)$.