**1.** Find the index of the smallest number in a sorted array, where the first $k$ numbers were shifted to the end.

1. How would you find the index of the smallest number.

   A brute-force solution is to ignore the sorted array property and scan every element in the array to find the smallest element. This is $\Theta(n)$.

   An algorithm that uses the divide-and-conquer principle is to first start from the leftmost index and the rightmost index as our list of candidates. Compare the middle element of that interval with the $A[right]$. If $A[right] < A[mid]$ then since $A[right]$ is the largest element in that subarray we cut the candidate size fo $[mid+1, right]$. else $A[mid] < A[right]$ then that subarray could not contain the smallest element and we reduce the size to candidate size to $[left, mid]$.

2. Write the pseudocode for the best algorithm you came up with.

3. Implement your answer using any programming language you want.

```
#include <iostream>
#include <vector>

using namespace std;

/**
 * T(n) = T(n / 2) + O(1)
 * T(n) = O(log_2(n))
 */
int min_element_in_rotated_array(const vector<int>& A){
    int left = 0, right = A.size() - 1;
    while(left < right){
        int mid = (left + right) / 2;
        if(A[mid] > A[right]){
            left = mid + 1;
        } else {
            right = mid;
        }
    }
    return left;
}
```

4. What is the time complexity of your answer?

   The recurrence is of the form $T(n) = T(\frac{n}{2}) + O(1)$ which evaluates to $T(n) = O(log_2(n))$.