a) $n = r - p + 1$, which is the total number of elements in array A. The worst-case behavior for RANDOMIZED-QUICKSORT occurs when the randomized-partition routine produces one subproblem with n-1 elements and one with 0 elements. And assume this unbalanced partitioning arises in each recursive call, then the partitioning costs $\Theta(n)$ time. Since the recursive call on an array of size 0 just returns, $T(0)= \Theta(1)$, and the recurrence for the running time is $T(n) = T(n - 1) + T(0) + \Theta(n) = T(n - 1) + \Theta(n)$. Intuitively, if we sum the costs incurred at each level of the recursion, we get an arithmetic series, which evaluates to $\Theta(n^2)$. For this problem, since in the worse case, the problem gets only 1 element smaller for each recursive call, it would require a total of n recursive calls to finish. Hence, the worst-case depth of the call-stack for Randomized-Quicksort is n.

b) The Randomized-Partition algorithm is correct. So, in order to show that MyQuicksort is correct, use prove by induction on n, where $n = r - p + 1$. First need to prove the base case is correct, where $n = r - p + 1 = 1$, in this case, there's only one element in the array that needs to be sorted, but since there's only one element, then it's already sorted. So the base case holds. Next, need to prove the inductive step. Since Randomized-Partition is correct, it means that every element in the subarray to the left of the pivot is smaller than or equals to the pivot, and every element in the subarray to the right of the pivot is larger than or equal to the pivot. So for MyQuicksort, given some array with length n, which means it has n elements in the array, we show that if MyQuicksort correctly sorts the entire array of length k, then it would correctly sort the array with length n, where $k < n$. Since Randomized-Partition is correct, the subarray to the left of the pivot has $k < n$ elements, which by induction, we can say it's correctly sorted, and the subarray to the right of the pivot has $n - k - 1 < n$ elements, we can also say it's correctly sorted by induction. Hence, by using induction, we have proved that MyQuicksort(A,p,q-1) and MyQuicksort(A,q+1,r) have the correct result, as a result, the entire input array A, at the end of the MyQuicksort(A,p,r) will be correctly sorted, and the correctness of MyQuicksort has been proved.

c) $n = r - p + 1$, which is the total number of elements in array A. The algorithm uses Randomized-Partition. And when in the worst case, Randomized-Partition could produce one subproblem with n-1 elements and one subproblem with 0 element. Assume this unbalanced partitioning arises in each recursive call, the partitioning cost $\Theta(n)$ time in the worst case. Since the recursive call on an array of size 0 just returns, $T(0) = \Theta(1)$, and for this algorithm, within each recursive calls, it chooses the smaller subarray for the recursive calls only. So the recurrence for the running time is $T(n) = T(0) + \Theta(n)$. For this problem, in the worst case, the problem gets 1 element smaller for each recursive call, and it would require a total of n recursive calls to finish. Hence, the worst-case depth of the call-stack for MyQuicksort is $\Theta(n)$.