1. How many different minimum cuts are there in a tree with n nodes (i.e. n-1 edges)?
The number of minimum cuts is the number of edges.
n - 1

2. Let "output" denote the cut output by Karger's min cut algorithm on a given connected graph with n vertices, and let p = $\frac{1}{(n \text{ choose } 2)}$ Which of the following statements are true?
For hints on this question, you might want to watch the short optional video on "Counting Minimum Cuts".
p is the lower bound of the probability of a fixed minimum cut. Thus, for every graph G with n nodes, there exists a minimum cut (A, B) of G such that Pr[out = (A, B)]≥p. Also, for every graph G with n nodes and every min cut (A, B) of G, Pr[out = A, B)]≥p. In the one case where a graph has only two nodes and one edge, then the probability of finding a min cut is 1 and p is 1. Therefore, the third answer is "There exists a graph G with n nodes and a min cut (A,B) of G such that Pr[out = A, B)]≤p".
For every graph G with n nodes, there exists a min cut (A, B) of G such that Pr[out = (A, B)]≥p
For every graph G with n nodes and every min cut (A, B) of G, Pr[out = A, B)]≥p
There exists a graph G with n nodes and a min cut (A,B) of G such that Pr[out = A, B)]≤p

3. Let 0.5 < α < 1 be some constant. Suppose you are looking for the median element in an array using RANDOMIZED SELECT (as explained in lectures). What is the probability that after the first iteration the size of the subarray in which the element you are looking for lies is ≤ α times the size of the original array?
Since α is > 0.5 (and < 1), the subarray cannot be greater than or equal to α, so the upper bound is α. If α was 0.7 and n was 10, then n * α = 7. In order for the subarray to be ≤ αtimes the size of the original array, the pivot cannot be greater than the 7th element, or smaller than the 3rd element (both 1 -α), so the answer is 1 - (1 - α) - (1 - α) = 0.6 (here) = 2 *α - 1
2 * α - 1

4. Let 0 < α < 1 be a constant, independent of n. Consider an execution of RSelect in which you always manage to throw out at least a 1− α fraction of the remaining elements before you recurse. What is the maximum number of recursive calls you'll make before terminating?
The first call will have all n elements, then a fraction of 1 - α of the elements are discarded. So, in the first recursive call, we have AT MOST n - n(1 - α) = n*α elements. On the ith recursive call, that is n*α^i elements. If we stop at the ith call, then then (α^i)* n ≤ 1. → log → d = −log(n) / log(α).
- log(n) / log(α)

5. The minimum s-t cut problem is the following. The input is an undirected graph, and two distinct vertices of the graph are labelled "s" and "t". The goal is to compute the minimum cut (i.e., fewest number of crossing edges) that satisfies the property that s and t are on different sides of the cut.
Suppose someone gives you a subroutine for this s-t minimum cut problem via an API. Your job is to solve the original minimum cut problem (the one discussed in the lectures), when all you

can do is invoke the given min s-t cut subroutine. (That is, the goal is to reduce the min cut problem to the min s-t cut problem.)

Now suppose you are given an instance of the minimum cut problem -- that is, you are given an undirected graph (with no specially labelled vertices) and need to compute the minimum cut. What is the minimum number of times that you need to call the given min s-t cut subroutine to guarantee that you'll find a min cut of the given graph?

With s fixed and t varying across the other vertices. Then we are finished by picking the smallest cut, which means at most n - 1 calls.

n - 1