1. We have a counter (a wireless hardware device) which can receive and process two commands, namely “initialize” and “increment”, from remote devices. The “initialize” command works reliably and correctly initializes the counter to zero. Unfortunately, the “increment” command does not work reliably. Each time an “increment” command is received, the counter increments the current count by 1 with probability \( \frac{1}{2} \). (Thus, the count remains the same with probability \( \frac{1}{2} \).) If after initializing the counter, the counter receives a sequence \( S \) of “increment” commands, the count displayed by the device may not be equal to \(|S|\), the actual number of commands in \( S \). However, consider the following strategy:

Suppose the count displayed at the end of the sequence \( S \) is \( q \).
Return \( 2q \) as the value of \(|S|\).

Show that the expected value returned by the above strategy is indeed equal to \(|S|\). (20 points)

2. We are given an undirected graph \( G(V, E) \) and we want to assign one of three colors, namely Blue, Green or Red, to each node of the graph. We say that an edge \( \{u, v\} \in E \) is “well colored” if the colors assigned to nodes \( u \) and \( v \) are different. Unfortunately, obtaining a 3-coloring that maximizes the number of well colored edges is computationally intractable. (The corresponding decision problem is NP-complete, just like SAT.) Consider the following randomized coloring scheme:

For each node \( v \), choose a color independently and uniformly at random from the set of three colors.

Suppose \( e^{*} \) denote the maximum number of edges that can be well colored by any 3-coloring of \( G \). Prove that the expected number of well colored edges produced by the above randomized coloring scheme is at least \( (2/3)e^{*} \). (25 points)

3. (30 points total) A \( d \)-ary heap is like a binary heap, but (with one possible exception) each non-leaf node has \( d \geq 2 \) children instead of two children.

(a) Prove that the height of a \( d \)-ary heap with \( n \geq 2 \) nodes is \( \lceil \log_{d} [n(d - 1)] \rceil \). (10 points)

(b) Give an efficient algorithm for the Extract-Max operation on a \( d \)-ary heap. Your algorithm must run in \( O(d \log_{d} [n(d - 1)]) \) time. Your answer should consist of pseudocode for the algorithm and an analysis of its running time. (20 points)
4. Suppose each row of an $n \times n$ matrix $A$ consists of 0's and 1's such that in any row of $A$, all the 1's come before any 0's in that row. Assuming that $A$ is already in memory, design an algorithm with a running time of $O(n)$ (not $O(n^2)$) to output the index of a row of $A$ with the maximum number of 1's.

Your answer must include the following: (i) A clear description of the algorithm and (ii) an explanation of why the running time of the algorithm is $O(n)$. You won't receive any credit if your algorithm is incorrect or its running time is asymptotically worse than $O(n)$. (25 points)