Question 1

(a) We will show that finding a \( \rho \)-approximation to the minimum number of bad edges in a 3-coloring of any graph is NP-Hard. In particular, we will show that 3-COLORING \( \leq_P \rho \text{BEC} \), where \( \rho \text{BEC} \) refers to the problem of getting within a factor of \( \rho \) of the minimal bad edge coloring.

Suppose we have a black box that finds a \( \rho \) approximation to \( \text{Opt} \). In particular, if we input a graph for which the best coloring has \( \text{Opt} \) bad edges, then our black box returns some value in the range between \( \text{Opt} \) and \( \rho \cdot \text{Opt} \). Given a graph \( G \), we would like to use this black box to determine whether or not \( G \) is 3-colorable. Simply input \( G \) into the black box, and if the black box returns 0, return YES. If the black box returns any value other than 0, return NO. This completes our algorithm. Since our algorithm involves a single call to the black box and a constant number of additional operations, we have a polytime reduction.

Now we simply need to show that \( G \in 3\text{-COLORING} \) if and only if our algorithm returns YES. Suppose \( G \in 3\text{-COLORING} \). Then there is a way to color \( G \) so that there are no bad edges. Thus \( \text{Opt} = 0 \). So our black box returns a value between 0 and \( \rho \cdot 0 = 0 \). Therefore our black box returns 0, and hence our algorithm returns YES.

Suppose now instead our algorithm returns YES. By construction, this must be because our black box returned 0. But if the black box returned 0, that means it found a 3-coloring containing no bad edges. This means that \( G \) is 3-colorable. Since our reduction was polytime, this shows that it is NP-Hard to find a \( \rho \)-approximation to this problem, for any constant \( \rho \).

(b) Our algorithm for this problem will be very similar to the one we used for approximating \( \text{MAX 3-SAT} \). In particular, for each node, we will uniformly and independently at random color it red, green, or blue (i.e. each node has a probability of \( \frac{1}{3} \) of being assigned each color, and the random choice for each node is independent of the random choices for any other node or nodes). Our claim is that in expectation, this produces
a solution with at least $\frac{2}{3} Opt'$ good edges. In fact, we will prove a slightly stronger statement, namely we will show that in expectation we have exactly $\frac{2}{3} m$ good edges in
our coloring of $G$, where $m$ is the number of edges in $G$. Since $Opt' \leq m$, this suffices to prove our claim.

To prove our claim, let's define $X$ to be a random variable that counts the number of good edges in our random coloring. Define $X_i$ to be an indicator variable for edge $i$. In particular, $X_i = 1$ if edge $i$ is a good edge in our coloring, and $X_i = 0$ otherwise. Note that $X = X_1 + \ldots + X_m$. Since we are looking for $E[X]$, and we know by linearity of expectation that $E[X] = \sum_{1 \leq i \leq m} E[X_i]$, we will focus on the value of $E[X_i]$ for a general $i$. By the definition of expectation,

$$E[X_i] = \sum_{j=0}^{\infty} j \cdot Pr[X_i = j]$$

Since $X_i$ only takes on the values of 0 and 1, this simplifies to $E[X_i] = Pr[X_i = 1]$, namely the probability that edge $i$ is good in our coloring. If we consider how the two endpoints of a given edge are colored, we note that there are only 9 possibilities, each of which occurs with probability $\frac{1}{9}$. Exactly 3 of these result in edge $i$ being bad: both nodes are red, both are green, and both are blue. Thus the probability that edge $i$ is good is $1 - \frac{3}{9} = \frac{2}{3}$. Plugging this into our definition of $E[X]$ we have that $E[X] = \frac{2}{3} m$, as desired.

(c) To derandomize this algorithm, consider the following algorithm. Select any uncolored node, and look at all of its colored neighbors (if there are any). Color it the color which appears least frequently among its neighbors (or on a tie, select from any of the colors which appear least frequently). For example, it will never matter what color we pick for the first node, so let's say we pick red. If the next node we pick is a neighbor of the previously picked node, it will be colored either blue or green. If at some later stage we are considering a node $v$, and of the neighboring nodes that already have been assigned color 4 are red, 5 are green, and 9 are blue, we will color $v$ red. We repeat until all nodes are colored. Once a color is selected for a node, we do not change it. We claim that at least $\frac{2}{3}$ of the edges in this coloring are good edges.

Proof of claim. Let's think of our algorithm as doing a little accounting work as it goes along. In particular, imagine that every time it colors the second endpoint of an edge, it labels that edge “good” if the edge's endpoints have different colors, and “bad” otherwise. Observe that since we color every node exactly once, every edge will be given a label once and only once in this version of the algorithm. Also note that the number of edges labeled “good” are exactly what we would like to count, and
further, our accounting has not actually changed the solution our algorithm generates. Therefore it suffices to show that whenever we color an edge, at least $\frac{2}{3}$ of the labels we add are "good". But this is clear by the definition of our algorithm; since we always pick the color which is used by the fewest number of neighbors, at most a third of the edge we label will be bad, and the rest will be good (this is a simple application of the pigeon-hole principle).

Note that this proof is somewhat deceptive: If we had tried to prove instead that for every node, at most a third of its neighbors are the same color, we would have failed, as this does not need to be the case. Instead, we must argue that as we add nodes, we are always creating twice as many good edges as bad edges.