Advanced Algorithms. Assignment 1

In the following exercises, Δ denotes the largest degree of the nodes in a given graph. That is, every node is assumed to have at most Δ neighbors. We assume Δ to be fixed.

Exercise 1.
An independent set in a graph is a subset X of nodes such that no edges exist between any two nodes of X. The Maximum Independent Set problem asks to find an independent set of maximum size in a given graph. This fundamental problem is known to be NP-complete.
Consider the following rather intuitive greedy algorithm: We take any node, put it in the solution, and remove this node and all its neighbors from the graph. We iterate this step until the graph is empty.

Prove that this algorithm always returns an independent set of size at least $1/\Delta$ of the maximum possible size. (By the way, this is also the best possible guarantee for this algorithm.)

Some remarks and hints – you may ignore this paragraph:
Specifically you must prove: If X is the greedy solution, and Y is an arbitrary independent set, then $|X| \geq |Y|/\Delta$. It should be rather easy to show $|X| \geq |Y|/(\Delta+1)$, which is acceptable as a “weak” submission. The stronger result $|X| \geq |Y|/\Delta$ is a bit harder to show. Working with upper and lower bounds is probably not helpful here. Instead you might cleverly “assign” certain nodes of X and Y to each other, and then study how the sizes are related.

Exercise 2.
A dominating set in a graph is a subset D of nodes such that every node is in D or has at least one neighbor in D. The Dominating Set problem asks to find a dominating set with a minimum number of nodes in a given graph. (Applications in networks include: routing, guarding, identifying hubs.) The problem is NP-complete, a fact that you need not prove here.

Propose an approximation algorithm that returns a dominating set being only $O(\log \Delta)$ times larger than a minimum solution.

You need not re-invent the wheel. It suffices to properly apply an already known algorithm and its properties. Nevertheless be precise enough in the details: How does your resulting algorithm work? Explain why it correctly returns a dominating set of the claimed size.