Section 1

The most important thing when writing software is thinking. A whole lot of code gets written before anyone bothers to think. And as a result, a whole lot of people end up fixing and then replacing what should have been thought of in the first place. Everything else you've listed is secondary. Within those, I would rank them: 1. Clarity. If your code is not clear, then the rest of us cannot understand it and maintain it. If we cannot maintain it, then it is utterly useless. It will have to be redone. Why even bother? No code is perfect. All code needs to be maintained. 2. Correctness. Obviously, if you code is doing the wrong thing, that's bad. I rank correctness second because if the code is clear and correct, then things below aren't significant.

Section 2

A robustness issue is easier to fix than a significant correctness issue. 3. Robustness. If you code falls over at the first challenge, that's just not good. 4. Efficiency. Your code should not waste resources. Everything is finite, even if it seems large. Waste not, want not. 5. Speed. Now we're so far down the list that we're talking fine nits. Speed is extremely important for very, very, very little code. For most code, it doesn't matter a whit. Don't bum instructions unless you absolutely have to. It usually is a hit on clarity and maintainability and thus is the wrong thing to do. 6. Brevity. All other things being equal, being unnecessarily wordy is not helpful. Clarity is foremost, but once you are crystal clear, then more is not helping. Stop, before you create contradictions and confusion. 7. Elegance. Elegance almost always is anti-clarity. Show me elegant code that doesn't require more explanation than the less elegant code and I'm in, but that's very, very rare. Don't let elegance become obfuscation.