Michael Xavier Canonizado
BSCS 3A - Software Engineering 1
Exercise 1

1. Studies show that maintenance and evolution costs can reach 80% or more of a software's total costs over its lifecycle. Explain why this value is so high.

As systems evolve, the codebase grows; and as the codebase grows, the number of dependencies and complexity increases. This then results in more code needed to maintain. New features require testing, enhancements, and bug fixes, and constant monitoring to be stable.

Another big factor to this is the age of the software. As the software ages, the tech usually becomes outdated, so the team either needs to migrate to a newer tech, which costs more and requires a lot of effort; or keep the existing but outdated tech and maintain the legacy code, which also cost more to find capable but expensive engineers to maintain the old architecture of the codebase.

2. In testing, there is this famous quote, by Edsger W. Dijkstra: "tests show the presence of bugs, but not their absence." Why are tests unable to show the absence of bugs?

This quote refers to how a test can only prove the robustness of a program in a particular case, given a set of parameters. But this cannot confirm other failures. A feature can have lots of unbounded and unanticipated amount of bugs, which can only be seen during actual user usage. Hence the quote "Tests show the presence of bugs,but not their absence".

3. In software project management, there is an empirical law, called Brooks' Law, which says that: "adding new devs to a project that is late, makes it even later." Why does this fact tend to be true?

This law highlights the challenge of hiring new developers into an existing codebase. Some might think that "more developers = faster development", but this can actually make development take longer due to the added overhead of hiring more developers. New developers tend to:

Need onboarding time. They need to understand the codebase, tools, and architecture to actively contribute to the project. This usually lasts a month or two, and at this time they are not really productive. So they are essentially being paid to understand the system.

Create negative contributions. Hiring inexperienced developers or even poorly onboarded developers may introduce bugs to the codebase which in turn requires rework or fixing, which further slows the progress.

Coordination. As the number of developers increases, the coordination and communication of contributors becomes less efficient. Adding new members can also disrupt the existing team dynamics, norms, and roles they may have, causing less productivity.

Therefore, when a project is already in its later stages, adding more developers may introduce more delays and further slow down development.

4. In 2015, it was discovered that millions of cars manufactured by a major automobile company emitted pollutants within legal standards only during laboratory tests. Under normal usage conditions, the cars released higher levels of pollutants to enhance performance. Thus, the code possibly included a decision command like the following one (merely illustrative). What would you do if your manager asks you to write an if like the one above?

This fundamentally is an ethical question which is addressed by the IEEE and ACM's code of ethics. We developers should be contributing to the benefit of society and well-being. In such cases where my manager is instructing me to provide a cheat-mode to bypass regulations is an epitome of a violation of this guideline. I would refuse the request and instead advocate for a better alternative that could increase performance but mitigates pollutant emissions.