1. **What’s wrong with just testing that a program works as expected?**

Testing that a program works as expected is important but not sufficient for ensuring the software's overall quality.

- Limited Scope: Focusing solely on whether the program works as expected often neglects edge cases or less obvious failure modes. It doesn't cover situations where users might behave unexpectedly or errors may arise due to unforeseen conditions.
- Overconfidence in Results: Even if the software passes all tests, it doesn't guarantee the absence of bugs or defects in areas that weren't tested. This can lead to undetected issues during real-world use.
- Missing Requirements Verification: Testing only the expected outcomes doesn't verify whether the software meets all specified requirements or user needs. Verification is crucial to ensure alignment with the original design and specification.

2. **How much more does it cost to fix a bug found after the product is released than it does from the very start of the project?**

Fixing a bug found after the product is released typically costs significantly more than fixing it during the earlier stages of the project. According to general industry studies, the cost of fixing a bug increases exponentially the later it is discovered in the software development life cycle.

- During Requirements Phase: If a bug is found early, while gathering requirements or defining specifications, the cost is minimal. The issue can be corrected before any coding begins.
- During Development Phase: If the bug is discovered during development (coding or unit testing), it still costs relatively little since changes only need to be made to the codebase.
- During Testing Phase: Bugs found in the testing phase (before release) are more expensive because they might require code refactoring, retesting, and additional verification.
- After Release (Production): If the bug is found after the product is released to users, the cost skyrockets. Fixing the bug may involve:

- Patch development and testing.
- Disruption to the end users or business operations.
- Customer support costs.
- Potential damage to the company's reputation.
- Possible financial or legal liabilities if the bug causes significant failures.

3. **What's the goal of a software tester?**

- Verify Requirements: Ensure that the software performs according to the specifications and meets the needs of its users and stakeholders.
- Identify Defects: Uncover bugs, errors, or failures in the software that could cause it to behave unexpectedly or fail in different scenarios.
- Ensure Quality: Evaluate the software for functionality, performance, usability, security, and reliability to ensure it provides a high-quality user experience.
- Prevent Future Issues: By identifying and resolving defects early, testers help prevent costly bugs from reaching production, reducing the risk of post-release failures.

4. **Give several reasons why the product specification is usually the largest source of bugs in a software product?**

- Ambiguity in Requirements: Specifications can be written in a way that leaves room for interpretation. Different developers, testers, or stakeholders may understand the same requirement differently, leading to inconsistencies in the implementation.
- Incomplete or Inaccurate Requirements: Product specifications may lack critical details or miss important edge cases, leading developers to make assumptions. These gaps often result in bugs when the software does not handle situations that weren't considered during specification.
- Changing Requirements: Requirements can evolve over time due to changing business needs, market demands, or user feedback. In such cases, if the specification isn't updated or tracked properly, it may lead to misaligned development and testing efforts, causing bugs.

- Miscommunication: Poor communication between stakeholders (e.g., clients, developers, and testers) can result in misunderstandings of the specifications. This often leads to software that does not meet the true needs or expectations of users, even if it follows the written specification.
- Lack of User Perspective: Specifications may not fully capture the real-world scenarios or user behavior. If the spec doesn't align with how users actually interact with the product, bugs can arise when the software behaves differently from user expectations.