1. **What are some common causes of flakiness in automation tests and what can you do to help mitigate these causes?**

Some of the common causes for flaky automation tests are,

* Poor construction of test framework/cases
* Tests not starting in a familiar state
* Broken environments and shared environments
* Test failures not managed very well
* Wait times

To mitigate these problems testers need to make sure develop some standard rules so issues won't be introduced if in case if we find any issues it will be easy to address in later stages.

* Test cases and libraries should follow a convention (naming, comments, etc.) that makes it easy to identify the scope of the test and keep it simpler.
* Always tests should start with the familiar state and end with cleanup so next test can start with a clean config (this will avoid unnecessary failures)

**For example**, if the environment is broken or shared/used by someone else auto test environment should be capable to handle this situation correctly and come out of this situation and run the tests.

* Tests should have the ability to verify the environment state and reset the environment state (if the environment is not in the expected state)

**For example**, when running a test which is using one device and that device is been reserved/used by someone else in the team the autotest should be able to end his session and set the environment to expected state and execute tests.

* Wait time should be well defined otherwise this can be some kind of issue in the autotest, some components can take longer to load so in these cases the test should be able to wait until the component has loaded/detected.

1. **Consider this scenario: You have estimated that a project needs 100 test cases automated, and it will take you 2 months to complete. However, you have now been told that you only have 1 month to complete the project. What options can you consider to solve this problem?**

Firstly, this can happen with various reasons.. but individually if im in this stage I will give some time to know the work and determine what can be done in this time frame considering these by asking my self these questions..

* Which functionality is most critical to the project?
* Which functionality is most noticeable to the user?
* Which functionality can be the highest security impact?
* Which functionality has the biggest business result on users?
* Which features of the application is most important to the end customer?
* Which features might have the most effect in the previous projects and caused problems?
* What do the developers think are the highest risk features of the application?
* How could we easily cover multiple functionalities?
* Which tests will have the best high risk coverage in a short time?

1. **Choose 3 software quality characteristics that relate to automation code and explain why you think they are important or not important to observe.**

**Reliability**

Reliability is the most valuable and commonly used in automation testing because a test can be failed with many reasons in the middle of a test run. When the test has been failed with a reason the system should be able to recover and continue the test run up to finish line.

For example, if the network goes down for 20 seconds then comes back the system should be able to recover and continue running the tests.

**Flexibility and Extensibility**

Flexibility and Extensibility will provide the ability to add, modify and remove functionality to test scripts whenever needed without breaking the current code system.

This is one of the most important characteristics in code development, because functionality changes may occur according to altering requirements, or an obligation if the development process is one of the iterative methods. Change is unavoidable in software development and so, this is one of the most important characteristics.

**Maintainability**

Maintainability is the principle that, because the code you write will be changed over time, maybe by yourself and perhaps by others, the code should be well structured, documented, and written using standard principles and frameworks.

Maintainability is a principle that is sometimes in conflict with other principles like optimization.

If this is not been followed when developing the code there are high chances might spend a lot of time debugging and to make changes. If followed correctly, you only need to modify the minimum possible number of existing test cases when the application changes. Building maintainable test cases also increase stability and make debugging easier.