1. What is Maven and Gradle?

Answer: Maven:

Definition: Maven is a build automation and project management tool used primarily for Java projects.

It helps manage dependencies, build processes, and project configurations.

Key Features: Centralized dependency management, standardized project structure, and a

plugin-based architecture.

Gradle:

Definition: Gradle is a build automation tool that combines the best features of Ant and Maven.

It uses Groovy-based DSL for build scripts and supports multiple programming languages.

Key Features: Flexibility in defining tasks and dependencies, incremental builds, and compatibility

with existing Maven and Ant projects.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2. What is Maven build lifecycle?

Answer: Maven build lifecycle is a sequence of phases that define the process of building and

managing a project. It consists of three standard lifecycles:

1. Clean Lifecycle:

clean: Removes artifacts and files generated by previous builds.

2. Default Lifecycle:

validate: Validates the project structure and dependencies.

compile: Compiles the source code.

test: Runs tests on the compiled code.

package: Packages compiled code into distributable format (e.g., JAR).

install: Installs the packaged artifact in the local repository.

deploy: Copies the packaged artifact to a remote repository.

3. Site Lifecycle:

site: Generates project documentation and reports.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3. What is a Maven repository?

Answer: A Maven repository is a storage location for Java libraries and plugins used in

Maven projects. It stores binary artifacts, such as JAR files, along with metadata about

those artifacts. Maven repositories can be local (on the developer's machine), central

(public repositories like Maven Central), or remote (custom repositories). They facilitate

dependency management by providing a centralized location for sharing and retrieving

libraries during the build process.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4. What is Maven artifact?

Answer: In Maven, an artifact is a deployable or distributable component of a project,

typically a JAR, WAR, or EAR file. Artifacts are generated during the build process and

can include compiled code, resources, and metadata. Maven uses artifacts to manage

dependencies, deploy applications, and share reusable components through repositories.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5. What is POM?

Answer: POM stands for Project Object Model in Maven. It is an XML file named pom.xml

that defines the configuration and metadata for a Maven project. The POM file contains

information about project dependencies, build settings, plugins, and other project-related

configurations. It serves as the central configuration file for Maven projects, guiding the

build process and managing project metadata.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6. How does Maven manage dependencies?

Answer: Maven manages dependencies by specifying them in the project's pom.xml file.

It uses a local repository to cache downloaded dependencies and searches Maven

Central for commonly used libraries. Maven automatically resolves transitive dependencies

and employs plugins to configure the build proces

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7. What is Maven plugin?

Answer: A Maven plugin is a set of tasks or goals that extends Maven's functionality.

Plugins are configured in the pom.xml file and perform various tasks such as compiling

code, running tests, packaging artifacts, and more.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

8. What Is Dependency Exclusion?

Answer: Dependency exclusion in Maven is a mechanism to exclude specific transitive

dependencies that are pulled in by a project's direct dependencies. By using the <exclusions>

element in the pom.xml file, developers can specify which transitive dependencies should be

excluded. This helps in managing the dependency tree and avoiding conflicts or unwanted

dependencies in the project.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

9. What is mojo?

Answer: In the context of Maven, a Mojo (short for "Maven plain Old Java Object") is a single

goal or task within a Maven plugin. It represents a unit of work that can be executed during

the build process. Mojos define the specific functionality of a plugin, and multiple mojos can

be part of a single plugin. Developers configure and use mojos in the pom.xml file to customize

and extend the Maven build process.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10. What Are The Maven Phases?

Answer: Maven phases represent different stages of the build lifecycle. Common Maven phases include:

1. validate: Validates the project structure and dependencies.

2. compile: Compiles source code.

3. test: Runs tests on compiled code.

4. package: Packages compiled code into a distributable format.

5. install: Installs the packaged artifact in the local repository.

6. deploy: Copies the packaged artifact to a remote repository.