1. What is Maven and why is it used?

Maven is a build automation tool primarily used for Java projects. It simplifies the build process by:

- Managing project dependencies.
- Automating the compilation, packaging, and deployment of code.
- Providing a standardized project structure.
- Integrating with CI/CD pipelines for automation.

2. Explain the POM file in Maven.

The POM (Project Object Model) file (pom.xml) is the central configuration file in Maven. It contains:

- Project metadata (groupId, artifactId, version).
- Dependencies to be included in the build.
- Build settings (plugins, goals, and properties).
- Repositories for dependencies and plugins.

3. What are Maven coordinates and what do they represent?

Maven coordinates uniquely identify a Maven artifact. They consist of:

- **groupId**: The group or organization (e.g., org.apache.maven).
- **artifactId**: The name of the artifact (e.g., maven-core).
- **version**: The artifact's version (e.g., 3.8)

4. How do you manage dependencies in Maven?

Dependencies are managed in the pom.xml file under the <dependencies> section. You define:

- The artifact's groupId, artifactId, and version.
- Optional scopes (e.g., compile, test, runtime).

5. What is a Maven repository and what are its types?

A Maven repository is a location where dependencies and plugins are stored. Types include:

- **Local repository**: Stored on the developer's machine.
- **Central repository**: Provided by Maven (default public repository).

6. Explain the concept of Maven lifecycle phases.

The Maven lifecycle consists of phases that define steps in the build process:

1. **validate**: Validates the project structure.
2. **compile**: Compiles source code.
3. **test**: Runs unit tests.
4. **package**: Packages the compiled code (e.g., JAR/WAR).
5. **install**: Installs the artifact into the local repository.
6. **deploy**: Deploys the artifact to a remote repository.

7. What are Maven goals and how do they differ from phases?

- **Phases**: High-level steps in the build lifecycle (e.g., compile, test).
- **Goals**: Specific tasks bound to a phase (e.g., compiler:compile). Goals can be executed independently of phases.

8. How do you create a Maven project?

1. Specify the groupId, artifactId, and version.
2. Maven generates a standard directory structure.

9. What is a Maven plugin and how is it used?

Plugins extend Maven's functionality. Examples:

- **Compiler plugin**: Compiles Java code.

Plugins are external features that are added or installed for better performance of the tools. Without some sort of plugins it is impossible to build the build process. It involves installing many plugins.

10. How do you handle versioning in Maven projects?

11. Explain the PEM file in Maven.

Pem file stored with the .pem extension. Usually it consists of Secure Shell key i.e., SSH key for running the instances. It is stored in the form of key value pair.

12. When do we generate JAR/WAR/EAR files in the target directory?

JAR/WAR/EAR files are generated in the target directory after running the package or later phases in the Maven lifecycle.

JAR files are created for the Java applications, WAR are created by building the Web bases applications and EAR files are created for the build of Enterprise Application Resources.

13. What is the home directory in Maven?

14. Where will the build files be stored?

Build files are stored in the target directory by default.

15. What is meant by a build tool?

A build tool automates tasks like compiling source code, packaging, running tests, and deploying applications.

16. Explain the process of building in Maven.

1. **Resolve dependencies**: Downloads required libraries.
2. **Compile**: Converts source code into bytecode.
3. **Test**: Runs unit tests.
4. **Package**: Creates a distributable format (e.g., JAR, WAR).

17. Does Maven support all types of projects to build?

Maven is a java build automation tool. It is only for Java projects. Different build tools are available for different types of projects like Gradle.

18. What is the difference between compile & validate?

- **validate**: Checks the project structure and dependencies.
- **compile**: Converts source code to bytecode.

19. Can you create only one JAR file or can we create multiple? Explain.

You can create multiple JAR files using multiple modules in a multi-module Maven project. Each module can have its own pom.xml.

20. What is Git and why is it used?

Git is a **distributed version control system** used for:

- Tracking code changes.
- Collaborating with teams.
- Managing multiple versions of code.

21. Explain the difference between Git and other version control systems.

Feature	**Git**	**Other VCS (e.g., SVN, CVS)**
Architecture	Distributed.	Centralized.
Speed	Faster due to local operations.	Slower due to server dependency.

22. How do you initialize a Git repository?

git init

23. What is the purpose of the .gitignore file?

The .gitignore file specifies files and directories to be ignored by Git (e.g., build files, logs).

24. How do you stage changes in Git?

git add <file>

25. What is the difference between git commit and git commit -m?

- **git commit**: Opens an editor to write a commit message.
- **git commit -m**: Includes the commit message inline.

26. How do you create a new branch in Git?

git branch <branch_name>

27. What is the difference between git merge and git rebase?

- **merge**: Combines branches, preserving history.

28. How do you resolve merge conflicts in Git?

- Edit conflicting files.
- Mark conflicts as resolved.
- Commit the changes.

29. What is the purpose of git stash?

Temporarily saves uncommitted changes, allowing you to switch branches.

30. Explain the use of git pull and git fetch.

- **git pull**: Fetches changes and merges them.
- **git fetch**: Fetches changes without merging.