🔧 Project-Based Question 1: Team Collaboration Setup

Scenario:
You're part of a new development team. Your task is to **set up Git for a new project**, configure global user settings, initialize the repo, and add all team members' `.gitignore` files to prevent tracking unnecessary files (like `.log`, `node_modules`, etc.).

Tasks:

- Set Git username and email globally.
- Initialize a Git repository in your project folder.
- Create and configure a `.gitignore` file.
- Add and commit the initial project structure.

Expected Concepts:
✅ Setting up Git
✅ Version Control basics
✅ `.gitignore`, `git init`, `git add`, `git commit`

🔧 Project-Based Question 2: Feature Development Workflow

Scenario:
You are building a new feature on a project and want to ensure a clean development process.

Tasks:

- Create a new branch named `feature/login`.
- Make changes to implement the login UI.
- Stage and commit your changes.
- Merge the feature branch into the main branch.
- Resolve any merge conflicts if they occur.

Expected Concepts:
✅ Branching
✅ Committing
✅ Merging
✅ Conflict Resolution

🔧 Project-Based Question 3: Rollback and Recovery

Scenario:
After deploying a new feature, the team finds a major bug. You need to **rollback the last commit** without deleting the changes and then fix the issue.

Tasks:

- Undo the last commit, preserving the changes locally.
- Modify the code to fix the bug.
- Stage, commit, and push the fixed version.

Expected Concepts:
✅ `git reset --soft HEAD~1`
✅ Staging and committing
✅ Amending commit history

🔧 Project-Based Question 4: Repository Cloning and Contributions

Scenario:
You're contributing to an open-source project. You need to **fork and clone a GitHub repo**, make changes, and prepare it for contribution.

Tasks:

- Clone the remote repository.
- Create a new branch for your changes.
- Add a new utility function in a file.
- Commit your changes with a proper message.
- Push your changes to the origin.

Expected Concepts:
✅ Cloning
✅ Branching
✅ Remote operations (`git push`, `origin`)
✅ Commit messages

🔧 Project-Based Question 5: Repository Audit and Clean-up

Scenario:
You are the release manager of a project. During a final code review, you notice some unnecessary

log files and test output files were accidentally committed. You want to **audit the Git history and clean the repository** by removing these files and updating the `.gitignore` file accordingly.

Tasks:

- View the commit history using `git log`.
- Identify and remove the last commit using `git reset`.
- Add a rule to `.gitignore` to exclude all `.log` files.
- Remove all `.log` files from the repository using Git commands.
- Commit the cleaned state with a proper message.

Expected Concepts:
✅ `git log`
✅ `git reset`
✅ `.gitignore` usage
✅ `git rm`
✅ `git commit`