**Aim 1: Clone, Create, Commit, and Push to Remote Question:**

1. Clone a remote Git repository to your local system.

git clone <https://github.com/Akm0222/akmvibes.git>

cd akmvibes

2. Create a new branch named feature-branch and switch to it.

git checkout -b feature-branch

3. Create a new file, newfile.txt, and add some content to it.

echo "This is a new file." > newfile.txt

git add newfile.txt

4. Stage and commit the changes with an appropriate commit message.

git commit -m "Add newfile.txt its Akanksha"

5. Push the feature-branch to the remote repository (GitHub)

git push origin branch name (feature-branch)

**Aim 2: Create and Merge Branches Question:**

1. Clone a repository and create a new branch named my-feature.

git clone <https://github.com/Akm0222/akmvibes.git>

cd akmvibes

git checkout -b feature-branch

2. Make a small change in an existing file (e.g., README.md) on the my-feature branch.

echo "Updated from my-feature branch" >> README.md

3. Stage and commit your changes.

git add README.md

git commit -m "Update README.md"

4. Merge the my-feature branch into the main branch locally after reviewing the changes

git checkout main

git merge feature-branch

If there are no branches or error, check by

git branch

Names of branches will appear

If no branch is seen create it with the command

git checkout -b branchname (main) // if main not present

**Aim 3: Revert a Commit Question:**

1. Clone the repository and make several commits (at least 3).

git clone <https://github.com/Akm0222/akmvibes.git>

cd akmvibes

echo "First change" > file.txt or revert.txt

git add file.txt

git commit -m "First commit"

echo "Second change" >> file.txt

git add file.txt

git commit -m "Second commit"

echo "Third change" >> file.txt

git add file.txt

git commit -m "Third commit"

2. Revert a specific commit from your commit history without removing history (use git revert).

git log --oneline

git revert fbf3533(no is your value appeared on screen)

Error aayga

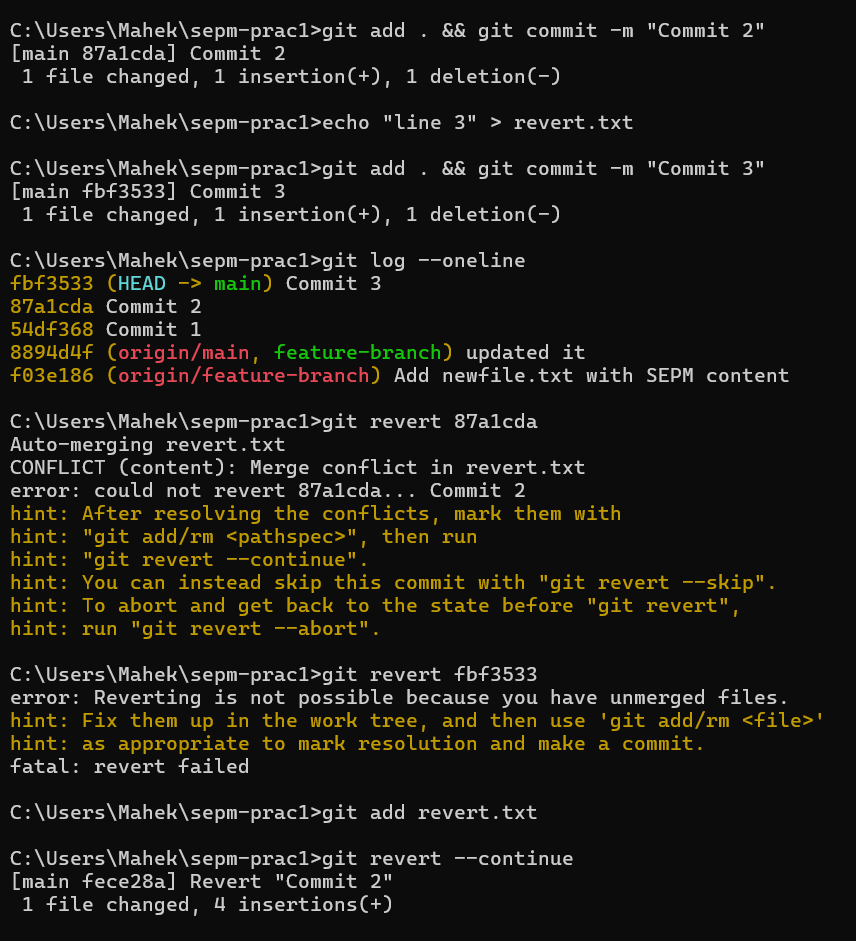
git add file.txt or git add revert.txt

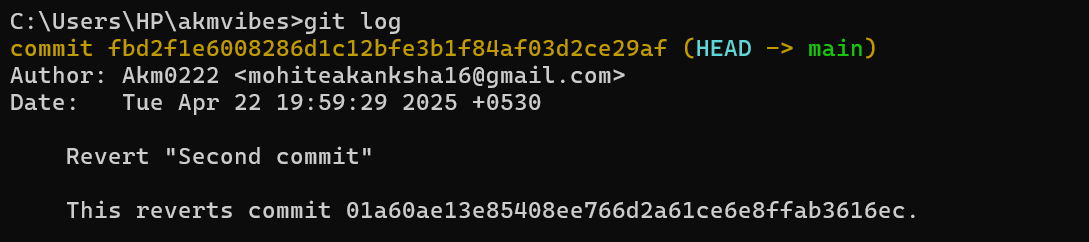
git revert --continue

(ajeeb si screen aaygi , press esc 2-4 times and type :wq)

3. Verify the revert and ensure a new commit is created that undoes the changes.

git log or git log --oneline





**Aim 4: Modify a Commit Message (Amend Last Commit) Question:**

1. Clone a repository and make a commit with an initial message.

git clone <https://github.com/yourusername/sepm-prac1.git>

cd sepm-prac1

Or

git clone <https://github.com/Akm0222/akmvibes.git>

cd akmvibes

2. Amend the commit message to correct any mistakes or improve the message.

echo "Initial content" > file.txt

git add file.txt

git commit -m "Init”

git commit --amend -m "Initial commit with correct message"

3. Verify the changes by checking the commit history.

git log --oneline

**Aim 5: Stash, Apply, and Pop Changes Question:**

1. Clone the repository and make uncommitted changes to an existing file.

git clone <https://github.com/yourusername/sepm-prac1.git>

cd sepm-prac1

Or

git clone <https://github.com/Akm0222/akmvibes.git>

cd akmvibes

echo "Some changes" >> file.txt

git add file.txt

2. Stash these changes temporarily.

git stash

3. Switch to a different branch and later return to the original branch.

git checkout feature-branch

git checkout main

4. Apply the stashed changes and commit them.

git stash apply

git add file.txt

git commit -m "Apply stashed changes"

OR

# Make changes

echo "Uncommitted change" >> README.md

# Stash the changes

git stash

# Switch to a different branch

git checkout -b another-branch

# Come back

git checkout main

# Apply and commit stash

git stash apply

git add README.md

git commit -m "Apply stashed changes"

**Aim 6: Resolve Merge Conflicts Question:**

1. Clone the repository and create two branches, branch-1 and branch-2.

git checkout -b branch-1

git checkout -b branch-2

2. Modify the same lines in the same file on both branches to create a conflict.

echo "Branch 2 change" > conflict.txt

git add conflict.txt

git commit -m "branch-2 change"

git checkout branch-1

echo "Branch 1 change" > conflict.txt

git add conflict.txt

git commit -m "branch-1 change"

3. Attempt to merge branch-2 into branch-1 and resolve the conflict manually.

git merge branch-2

# Resolve manually in an editor

git add conflict.txt

git commit -m "Resolve merge conflict between branch-1 and branch-2"

4. Commit the resolved file and verify the successful merge.

notepad conflict.txt

git add conflict.txt

git commit -m "resolve conflict"

OR

# Create branch-1 and edit file

git checkout -b branch-1

echo "Edit from branch-1" > conflict.txt

git add . && git commit -m "Edit in branch-1"

# Create branch-2 and edit same file differently

git checkout main

git checkout -b branch-2

echo "Edit from branch-2" > conflict.txt

git add . && git commit -m "Edit in branch-2"

# Try merging and resolve conflict

git checkout branch-1

git merge branch-2

# Open conflict.txt, manually resolve, then:

git add conflict.txt

git commit -m "Resolved conflict between branch-1 and branch-2"

HOW TO OPEN FILE MANUALLY?

notepad filename.txt

**Aim 7: Work with Git Submodules Question:**

1. Clone a repository with submodules.

git clone <https://github.com/yourusername/sepm-prac1.git>

cd sepm-prac1

Or

git clone <https://github.com/Akm0222/akmvibes.git>

cd akmvibes

2. Initialize and update the submodules.

3. Modify a file in one of the submodules and commit the change.

Go to repo

4. Commit the changes to the main repository and verify that the submodule changes are

Go to git

Make new repo named submodule-repo

U will see some blue link as creating a new file

* Name your file: subfile.txt
* Add content: e.g., Hello from submodule

Commit changes upar hi kahi hoga

On cmd:

git submodule update --init --recursive

cd submodule

echo "More content from reflected”.

echo "More content from Mahek" >> subfile.txt

git add subfile.txt

git commit -m "Update subfile.txt inside submodule"

cd ..

git add submodule

git commit -m "Track updated submodule version"

git push

git push --set-upstream origin branch-1 (if error )

**Aim 8: Cherry-pick a Commit Question:**

1. Clone the repository and make several commits on a different branch (e.g., feature branch).

git checkout -b feature-branch

echo "Feature 1" > cherry.txt

git add . && git commit -m "Add Feature 1"

echo "Feature 2" >> cherry.txt

git add . && git commit -m "Add Feature 2"

2. Identify a commit you want to cherry-pick from feature-branch to the main branch.

git log --oneline

3. Use git cherry-pick to bring that commit to the main branch and commit the changes

git checkout main

git cherry-pick <commit-hash-of-feature-1>

git clone https://github.com/yourusername/sepm-prac1.git

cd sepm-prac1

# Create feature branch and make multiple commits

git checkout -b feature-branch

echo "Feature 1" > cherry.txt

git add . && git commit -m "Add Feature 1"

echo "Feature 2" >> cherry.txt

git add . && git commit -m "Add Feature 2"

# Get commit hash of "Feature 1"

git log --oneline

# Switch to main and cherry-pick

git checkout main

git cherry-pick <commit-hash-of-feature-1>