Ex. No. 4	Advanced Git Commands
28/08/2024	

AIM

Algorithm for Advanced Git Commands

Algorithm

1. **Interactive Rebase**:
   - Review the commit history using `git log --oneline`.
   - Enter interactive rebase mode with `git rebase -i HEAD~N` (where N is the number of commits).
   - Modify, edit, or squash commits as necessary to clean up the history.
   - Rebase your branch onto another branch if needed using `git rebase <branch>`.
2. **Stashing Changes**:
   - Make changes in your working directory.
   - Stash changes using `git stash`.
   - Apply the stashed changes when required using `git stash apply`.
3. **Reverting and Resetting**:
   - Create a new commit and revert it using `git revert <commit_hash>`.
   - Experiment with `git reset --soft`, `--mixed`, and `--hard` to understand the differences in resetting commit history.
4. **Cherry-Picking a Commit**:
   - Select a commit from another branch and apply it using `git cherry-pick <commit_hash>`.
   - Resolve any conflicts that arise and continue the cherry-pick process.
5. **Working with Submodules**:
   - Add a submodule to your repository using `git submodule add <repository_url> <path>`.
   - Clone a repository with submodules using `git clone --recurse-submodules <repository_url>`.
   - Update submodules using `git submodule update --remote --merge`.
6. **Implementing Git Hooks**:
   - Create a pre-commit hook in the `.git/hooks` directory to prevent commits with "TODO" comments.
   - Make the script executable and test it by attempting to commit a file with a "TODO" comment.

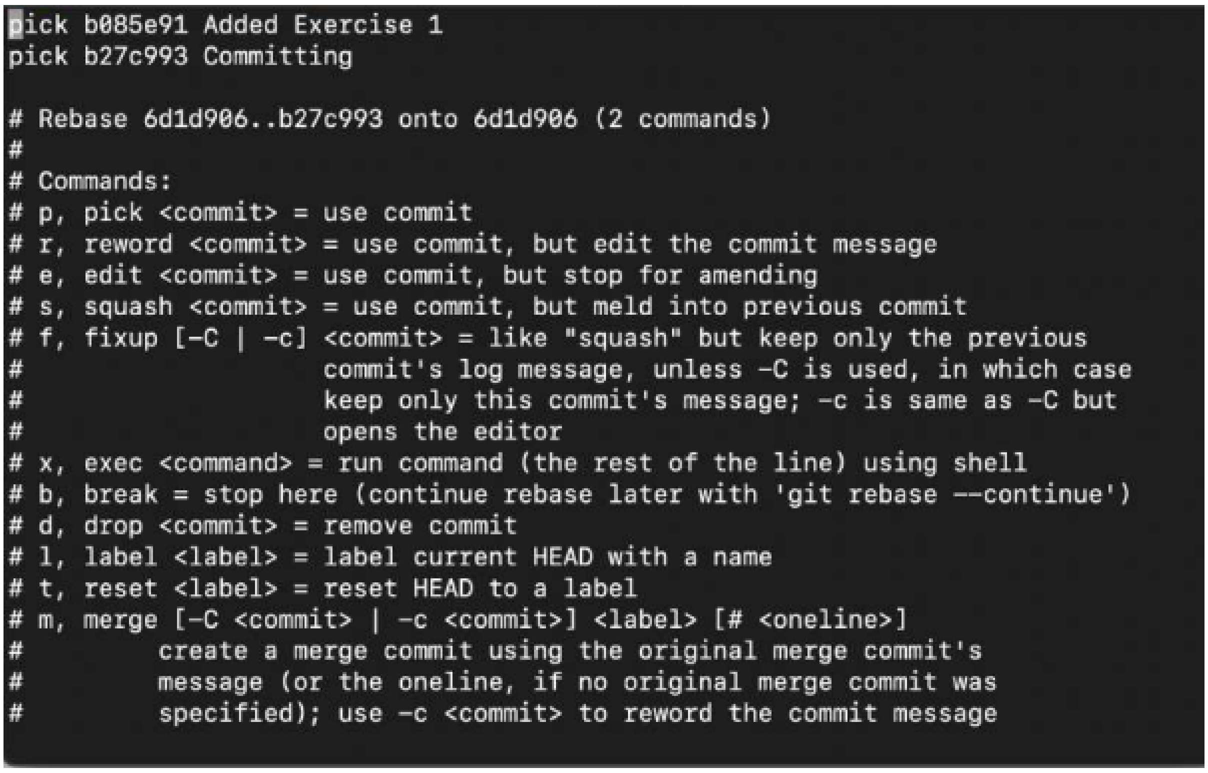
Input

```
# Task 1: Interactive Rebase
git log --oneline
git rebase -i HEAD~N
# Task 2: Stashing Changes
git stash
git stash apply
# Task 3: Reverting and Resetting
git revert <commit_hash>
git reset --soft <commit_hash>
git reset --mixed <commit_hash>
git reset --hard <commit_hash>
# Task 4: Cherry-Picking a Commit
git cherry-pick <commit_hash>
# Task 5: Working with Submodules
git submodule add <repository_url> <path>
git clone --recurse-submodules <repository_url>
git submodule update --remote --merge
# Task 6: Implementing Git Hooks
echo "#!/bin/sh
if grep -r \"TODO\" .; then
    echo \"Commit rejected: please remove TODO comments.\"
    exit 1
fi" > .git/hooks/pre-commit
chmod +x .git/hooks/pre-commit
```

Output

1. **Interactive Rebase**:

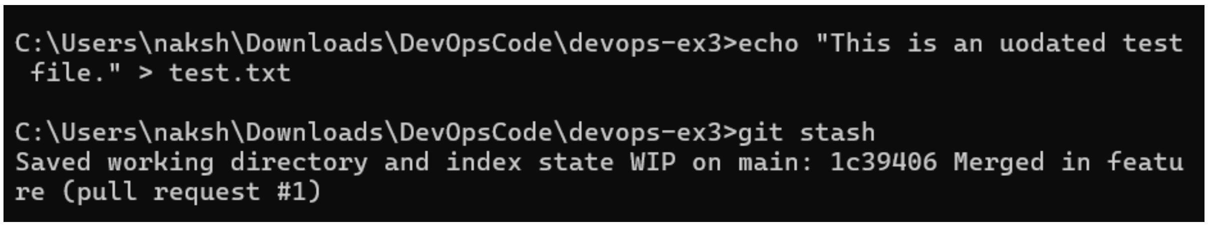
   - Successfully rebased the commit history to clean up and combine multiple commits.



```
pick b085e91 Added Exercise 1
pick b27c993 Committing

# Rebase 6d1d906..b27c993 onto 6d1d906 (2 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#         create a merge commit using the original merge commit's
#         message (or the oneline, if no original merge commit was
#         specified); use -c <commit> to reword the commit message
```

2. **Stashing Changes**:

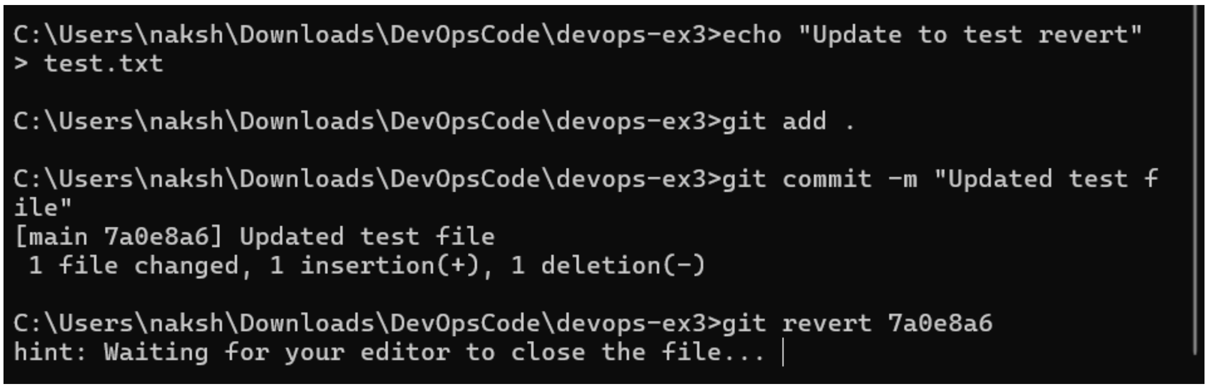
   - Successfully stashed and reapplied changes.



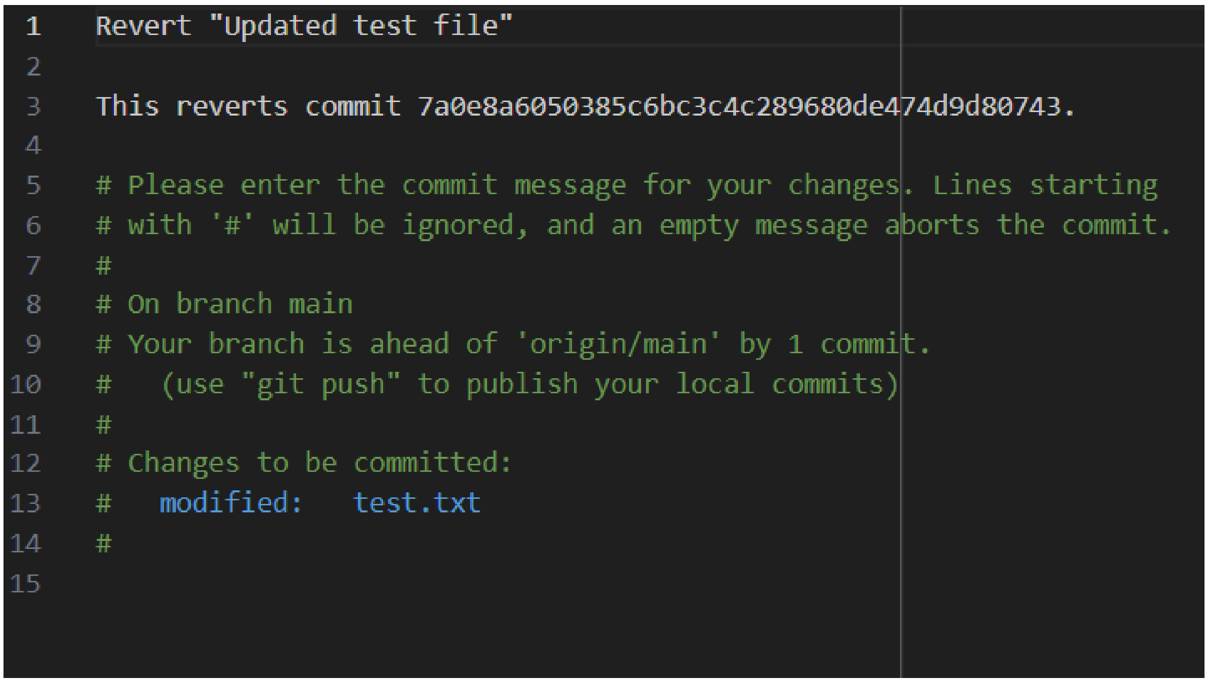
```
C:\Users\naksh\Downloads\DevOpsCode\devops-ex3>echo "This is an uodated test
 file." > test.txt

C:\Users\naksh\Downloads\DevOpsCode\devops-ex3>git stash
Saved working directory and index state WIP on main: 1c39406 Merged in featu
re (pull request #1)
```

3. **Reverting and Resetting**:

   - Successfully reverted a commit and experimented with different reset options.



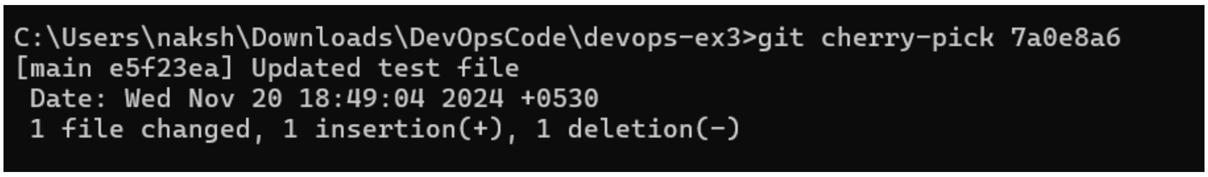
```
C:\Users\naksh\Downloads\DevOpsCode\devops-ex3>echo "Update to test revert"
> test.txt

C:\Users\naksh\Downloads\DevOpsCode\devops-ex3>git add .

C:\Users\naksh\Downloads\DevOpsCode\devops-ex3>git commit -m "Updated test f
ile"
[main 7a0e8a6] Updated test file
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\naksh\Downloads\DevOpsCode\devops-ex3>git revert 7a0e8a6
hint: Waiting for your editor to close the file...
```



```
Revert "Updated test file"

This reverts commit 7a0e8a6050385c6bc3c4c289680de474d9d80743.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is ahead of 'origin/main' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#   modified:   test.txt
#

```

4. **Cherry-Picking a Commit**:

   - Successfully cherry-picked a commit from another branch and resolved any conflicts.



```
C:\Users\naksh\Downloads\DevOpsCode\devops-ex3>git cherry-pick 7a0e8a6
[main e5f23ea] Updated test file
 Date: Wed Nov 20 18:49:04 2024 +0530
 1 file changed, 1 insertion(+), 1 deletion(-)
```

5. **Working with Submodules**:

   - Successfully added, cloned, and updated submodules in the repository.

6. **Implementing Git Hooks**:
    - Successfully created a pre-commit hook that prevents commits with "TODO" comments.

Result

By following this AIM Algorithm, you have successfully mastered advanced Git operations, including interactive rebase, stashing, reverting, resetting, cherry-picking, managing submodules, and implementing Git hooks. This exercise enhances your ability to manage commit histories effectively, collaborate efficiently, and enforce coding standards within your projects.