# Exercise: Handling Stashed and Lost Changes

This document defines advanced exercises focused on **stashing, restoring lost changes, and moving commits**. These tasks simulate real-world Git scenarios requiring deeper problem-solving skills.

## Problem 1: Stashing and Selective Restore

### Problem Description

A developer is working on multiple changes but needs to **stash only specific modifications** while keeping others.

### Task

1. Create a new branch and modify two files:
   * featureA.txt
   * featureB.txt
2. Stash only featureA.txt while keeping featureB.txt.
3. Verify that featureA.txt is stashed but featureB.txt remains.
4. Switch to another branch and make a change.
5. Apply only the specific stash without popping it.
6. Confirm that featureA.txt is restored.

## Problem 2: Recovering Uncommitted Changes

### Problem Description

A developer accidentally discards uncommitted changes and needs to **recover them**.

### Task

1. Create and modify a file:
   * critical.txt.
2. Discard the uncommitted changes.
3. Recover the lost changes.
4. Verify that critical.txt is restored.

## Problem 3: Moving Multiple Commits Between Branches

### Problem Description

A developer mistakenly commits multiple changes in the wrong branch and must **move them without merging everything**.

### Task

1. Create and modify two files:
   * file1.txt
   * file2.txt
2. Switch to the correct branch and cherry-pick multiple commits.
   * main branch
3. Verify that both commits are now in main.

## Problem 4: Stashing and Resolving Conflicts After Merge

### Problem Description

A developer needs to **stash changes, merge a branch, and resolve conflicts before applying stashed work**.

### Task

1. Create a branch and make conflicting changes:
2. Switch to main and modify the same file.
3. Switch back to conflict-branch, modify another file, and stash changes.
4. Merge conflict-branch into main, resolve the conflict, and complete the merge.
5. Apply the stashed changes and verify they were not lost.