Resolving merge conflicts after a Git rebase

When you perform a `git rebase` operation, you're typically moving commits around. Because of this, you might get into a situation where a merge conflict is introduced. That means that two of your commits modified the same line in the same file, and Git doesn't know which change to apply.

After you reorder and manipulate commits using `git rebase`, should a merge conflict occur, Git will tell you so with the following message printed to the terminal:

```
error: could not apply fa39187... something to add to patch A

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
Could not apply fa39187f3c3dfd2ab5faa38ac01cf3de7ce2e841... Change fake file
```

Here, Git is telling you which commit is causing the conflict (`fa39187`). You're given three choices:

- You can run `git rebase --abort` to completely undo the rebase. Git will return you to your branch's state as it was before `git rebase` was called.
- You can run `git rebase --skip` to completely skip the commit. That means that none of the changes introduced by the problematic commit will be included. It is very rare that you would choose this option.
- You can fix the conflict.

To fix the conflict, you can follow the standard procedures for resolving merge conflicts from the command line. When you're finished, you'll need to call `git rebase --continue` in order for Git to continue processing the rest of the rebase.