1. Which procedure does a better job of preserving a project's history?

1 / 1 point

[✓ **git merge**]

git rebase

✓ Correct

git merge preserves the full history and adds to it

2. Why might a merge result in problems that do not show conflicts? Select all answers that apply.

1 / 1 point

[✓ A merge might affect code in a very different part of the product in a non-obvious way and not receive enough testing]

✓ Correct

This answer explains itself

The merge may alienate a developer who was opposed to it

[✓ There may be two different solutions to the same problem which interfere with other]

✓ Correct

The code might look fine, but the results are problematic

It may increase the cost of the product

3. A **git rebase:**

1 / 1 point

[✓ Adapts a branch to incorporate the latest changes in another branch without yet merging this branch into the other branch]

Is not a legal operation in a public repository

Hides changes so no one can trace where they are coming from

✓ Correct

This is the purpose of a rebase

4. How would you merge two branches (**br1** and **br2**) into the main branch? **1 / 1 point**

- ○ **git checkout main && git merge br1 br2**
- ○ **git checkout br1 && git merge br2 && git checkout main && git merge br1**
- [✓ **git checkout main && git merge br1 && git merge br2**]

✓ **Correct**
You can only merge one branch at a time

5. What do you do when a merge fails? **1 / 1 point**

- [✓ Evaluate the conflict, see what the correct result should be and then fix]
- ○ Abandon all changes desired and start over with a **git revert**
- ○ Find out who is to blame and force them to fix their errors

✓ **Correct**
This is correct