Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

Each answer shows the steps, then the **result**.

2.1

commit: a snapshot of the files with its message, author and parent

branch: a name that points to the newest commit of a line of work

merge: a commit with two parents that joins two lines

2.2

say what changed: “Reject a 6th loan in LoanService”

and why, with a link: “library policy, story 14; adds test”

2.3

$\lceil \log_2 1000 \rceil$

10

2.4

rebase makes new commits with new IDs

teammates’ work still points to the old ones, so their next pull duplicates or loses work

rule: rewrite only your own unpublished commits

3.1

(a) both branches changed the same line in different ways, and git cannot know which is right

(b) find the requirement: story 14 sets the limit to 6; keep `6` and the useful new comment, remove the markers; run the tests before committing

(c) short branches merged at least daily; small pull requests, and saying in the team chat who is changing which module

(d) which choice was made and why: “limit 6 from story 14, kept the comment”

3.2

(a) with: $(1 + 2 + 96 + 1)/4 = \mathbf{25}$ h; without: $4/3 \approx \mathbf{1.3}$ h

(b) reviewers put it off and then check it less carefully; it collides with more of other people’s work, and bugs are harder to find in it

(c) keep pull requests under about 200 lines; split big features behind a switch (feature flag)

3.3

(a) a broken commit breaks everyone’s work at once, with no review

exercises.lol exercises.com.cn

(b) huge merge conflicts in the last week; features that work alone fail together, too late to fix

(c) a branch per small task, merged within a day or two; a pull request with one reviewer; `main` is protected: tests must pass and no force-push

(d) short branches with pull requests: conflicts stay small, every change is reviewed, and `main` always works for the demo