30538 Problem Set 3: git Peter Ganong, Maggie Shi, and Dema Therese Maria 2024-10-21 Due Sat Oct 26 at 5:00PM Central. Worth 50 points. Before you begin this problem set, you will need to install command-line git. Your submission does not need runnable code. Instead, you will tell us either what code you ran or what output you got. 1. Late coins used this pset: **1** 2. Late coins left after submission: **2**

Instructions This problem set has two parts: a Solo section and a Partnered section. Each section has its own GitHub repository and GradeScope submission. Please read their respective instructions carefully to ensure you follow the correct workflow for each part. SECTION 1 - Solo 1. Setup Instructions: • Each student must individually accept the solo repository from GitHub Classroom, and complete the solo exercises.

2. Submission Process: • "I have uploaded the names of anyone I worked with on the problem set here" (1 point) • Knit your ps3_solo.qmd as a pdf. • Push ps3_solo.qmd and ps3_solo.pdf to your github repo. Use command line git (not github desktop) • Submit ps3_solo.pdf through the Problem Set 3 Solo assignment on Gradescope. (4 points) • Tag your submission in Gradescope (applies only to the solo part) "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **DA** problem set setup (5 points)

Learn git branching (15 points) Go to https://learngitbranching.js.org. This is the best visual git explainer we know of. 1. Complete all the levels of main "Introduction Sequence". Report the commands needed to complete "Git rebase" with one line per command.

git branch bugfix git commit git switch bugfix git commit git rebase main

2. Complete all the levels of main "Ramping up". Report the commands needed to complete "Reversing changes in git" with one line per command.

git reset local git reset c1 git switch push git revert c2

3. Complete all the levels of remote "Push & Pull – Git Remotes!". Report the commands needed to complete "Locked Main" with one line per command.

Exercises (25 points) Now it's time to get your hands dirty! Clone d Tips: • These exercises have many steps. Keep a notebook (e.g. .txt or note-taking software) with what happens at every step. • To find out what directory you are in, run cd on a PC or pwd on a Mac. • Make sure you're navigating to the correct folder for each exercise. • When running git merge or git revert without a commit message, Git opens a VIM text editor by default. To avoid VIM, add the -m "commit message" option to the git merge or git revert command. If you find yourself stuck in VIM, type :q! and click Enter/Return.

2 Basic Staging and Branching (5 points) 1. Exercise. For your pset submission, tell us only the answer to the last question (22).

"PS C:-katas-staging> git status On branch master nothing to commit, working tree clean"

2. Exercise. For your pset submission, tell us only the output to the last question (18).

PS C:-katas-branching> git diff mybranch master
diff –git a/Daniel b/Daniel deleted file mode 100644 index 0226fd2..0000000 — a/Daniel +++ /dev/null

@@ -1 +0,0 @@ -Daniel Avila diff –git a/file2.txt b/file2.txt new file mode 100644 index 0000000..9a0301e — /dev/null +++ b/file2.txt @@ -0,0 +1 @@ +fun in the sun!

Merging (10 points) 1. Exercise. After completing all the steps (1 through 12), run git log –oneline –graph –all and report the output.

PS C:-katas-merge> git log –oneline –graph –all * 299edf9 (HEAD -> master) commit! * 095acc1 PPS C:-katas-branching> git diff mybranch master diff –git a/Daniel b/Daniel deleted file mode 100644 index 0226fd2..0000000 — a/Daniel +++ /dev/null @@ -1 +0,0 @@ -Daniel Avila diff –git a/file2.txt b/file2.txt new file mode 100644 index 0000000..9a0301e — /dev/null +++ b/file2.txt @@ -0,0 +1 @@ +fun in the sun!S C:-katas-branching> git diff mybranch master diff –git a/Daniel b/Daniel deleted file mode 100644 index 0226fd2..0000000 — a/Daniel +++ /dev/null @@ -1 +0,0 @@ -Daniel Avila diff –git a/file2.txt b/file2.txt new file mode 100644 index 0000000..9a0301e — /dev/null +++ b/file2.txt @@ -0,0 +1 @@ +fun in the sun! * 186e543 Add content to greeting.txt * c782d60 Add file greeting.txt

2. Exercise. Report the answer to step 11.

PS C:-katas\3-way-merge> git log –oneline –graph –all

- 2412988 (HEAD -> master) Merge branch 'greeting' so that we can have the two together | | * aa1170f (greeting) hey big fella commit:
- a903eb6 adding readme
  |/
- dea8084 Add content to greeting.txt
- fa7fef9 Add file greeting.txt

3. Identify the type of merge used in Q1 and Q2 of this exercise. In words, explain the difference between the two merge types, and describe scenarios where each type would be most appropriate.

The difference between the ff-merge and the 3 way merge is that the 3 way merge requires you to resolve two or more branches that have diverged from the main branch. The ff-merge is just a straight forward merge that does not require an additional verification step since the main branch is still similar with the branch you are currently working on.

Undo, Clean, and Ignore (10 points) 1. Exercise. Report the answer to step 13.

PS C:-katas-revert> git show 05d95 commit 05d95010a3025dc98182af8a1080dc42512dc44c Author: git-katas trainer bot git-katas@example.com Date: Sun Oct 27 08:17:17 2024 -0500

```
Add credentials to repository
```

diff –git a/credentials.txt b/credentials.txt new file mode 100644 index 0000000..8995708 — /dev/null +++ b/credentials.txt @@ -0,0 +1 @@ +supersecretpassword

2. Exercise. Look up git clean since we haven't seen this before. For context, this example is about cleaning up compiled C code, but the same set of issues apply to random files generated by knitting a document or by compiling in Python. Report the terminal output from step 7.

PS C:-katas-cleaning> git clean -f -d
Removing obj/

3. Exercise. Report the answer to 15 (“What does git status say?”)

PS C:-katas
> git status On branch master Changes to be committed: (use “git restore –staged …” to unstage) deleted: file1.txt

Changes not staged for commit: (use “git add …” to update what will be committed) (use “git restore …” to discard changes in working directory) modified: .gitignore

Untracked files: (use “git add …” to include in what will be committed) file1.txt file3.txt