Семинар #3: git, продвинутый. Практика.

Задачи с прошлого семинара

Решите новые задачи с прошлого семинара. Задача 1 и задача 12.

Подготовка. Генерация репозитория, который будет использоваться в задачах.

Для решения некоторых задач вам потребуется создать репозиторий коммитов. Его можно найти по адресу: mipt-hsse.gitlab.yandexcloud.net/v.biryukov/utils. Клонируйте этот репозиторий себе. Решите следующие задачи:

Задачи

1. **Просмотрите граф коммитов**:
   Это можно сделать на GitLab во вкладке `Code -> Repository Graph` или у вас на машине, с помощью команды:

   ```
   git log --oneline --all --graph
   ```

2. **Новый alias**:
   Добавьте новый `alias` для этой команды, назовите его `lg`. То есть теперь, при вызове:

   ```
   git lg
   ```

   Должно печататься то же, что и при вызове

   ```
   git log --oneline --all --graph
   ```

3. **Слияние с конфликтом**:
   Слейте ветку `bob` в ветку `main`. Для этого вам нужно перейти в ветку `main` и вызвать `git merge bob`. При этом могут возникнуть конфликты. Разрешите их. (в этой и последующих задачах необязательно, чтобы код после слияния был корректным, так как это задание на git, а не на язык программирования).

4. **Отмена слияния**:
   Отмените, сделанное только что слияние.

5. **Перебазирование с конфликтом**:
   Перебазируйте ветку `bob` на ветку `main`. При этом могут возникнуть конфликты. Разрешите их.

6. **Отмена перебазирования**:
   Отмените, сделанное только что перебазирование.

7. **Копия коммита с конфликтом**:
   Перейдите на ветку `main` и скопируйте в неё один коммит (`git cherry-pick`) с хэшем `893e28d` из ветки `bob`. При этом могут возникнуть конфликты. Разрешите их.

8. **Отмена `cherry-pick`**:
   Отмените, сделанный только что `cherry-pick`.

9. **Слияние быстрой перемоткой**:
   Перейдите на ветку `alice` и слейте в ветку `alice` ветку `main`, используя быструю перемотку.

10. **Отмена быстрой перемотки**:
    Отмените, сделанную только что быструю перемотку.

11. **Слияние без быстрой перемоткой**:
    Перейдите на ветку `alice` и слейте в ветку `alice` ветку `main`, без быстрой перемотки.

12. **Отмена последнего слияния**:
    Отмените, сделанное только что слияние.

13. **Разница**:
    Перейдите в ветку `casper` и найдите `diff` последних двух коммитов.

14. **Патч-файл**:
    Создайте патч-файл, содержащий `diff` последних двух коммитов ветки `casper`. Перейдите в ветку `main` и примените этот патч. Закоммите изменения в ветке `main`.

15. **Разница одного файла**:
    Напечатайте `diff` файла `integration.py` в ветках `main` и `casper`.

16. **git bisect**:
    Перейдите в ветку `casper` и запустите скрипт `sorting.py`

    ```
    python ./sorting.py
    ```

    Вы увидите, что одна из сортировок работает неправильно, хотя в других ветках эта сортировка работала правильно. Значит в одном из коммитов ветки `casper` була допущена ошибка. Найдите коммит, в котором была допущена ошибка с помощью `git bisect`. Исправьте эту ошибку.

17. **Интерактивный `rebase`**:
    Сделайте интерактивный `rebase` ветки `casper` с момента отсоединения её от других веток. Склейте вместе коммиты ветки `casper`, так чтобы там осталось 4 коммита.

18. **Отмена интерактивного `rebase`**:
    Отмените только что сделанный интерактивный `rebase`.

19. **git add -p**:
    Перейдите на ветку `casper` и добавьте изменения в рабочую папку и в область индексирования. Изменить функции `add`, `factorial` и `is_prime` в файле `arithmetic.py`, а также сортировку `bubble_sort` в файле `sorting.py`. Добавьте в индекс только изменения функций `add` и `is_prime`. Остальные изменения добавлять не нужно. Добавлять новый коммит не нужно.

20. **git stash**:
    После изменений, сделанных в прошлой задаче, попробуйте переключиться на ветку `main`. Произойдёт ошибка, так как есть изменения в рабочей директории и в индексе. Используйте `git stash`, чтобы спрятать изменения. Перейдите на ветку `main` и сделайте там какой-нибудь коммит на ваш выбор. Вернитесь на ветку `casper` и вытащите изменения из `stash`.

21. **Перенос git stash**:
    Снова добавьте эти же изменения в `stash`. Перейдите на ветку `bob` и вытащите изменения, сделанные на ветке `casper`, в ветку `bob`. Возможны конфликты. Сделайте коммит с этими изменениями в ветке `bob`.

Низкоуровневый git

В этом задании нельзя использовать команды, которые мы изучали прежде (кроме `git init`). Можно использовать только следующие низкоуровневые команды:

- git init
- git hash-object
- git cat-file
- git ls-files
- git mktree
- git commit-tree
- git update-ref
- git ls-tree

А также команды для проверки результатов:

- git status
- git log
- git show
- git diff

Создайте репозиторий, состоящий из 3-х коммитов и 2-х веток, используя только эти команды.

* Ещё более низкоуровневый git

Решите предыдущую задачу вообще без использования команд git. Можно только использовать команды для проверки результатов:

- git status
- git log
- git show
- git diff