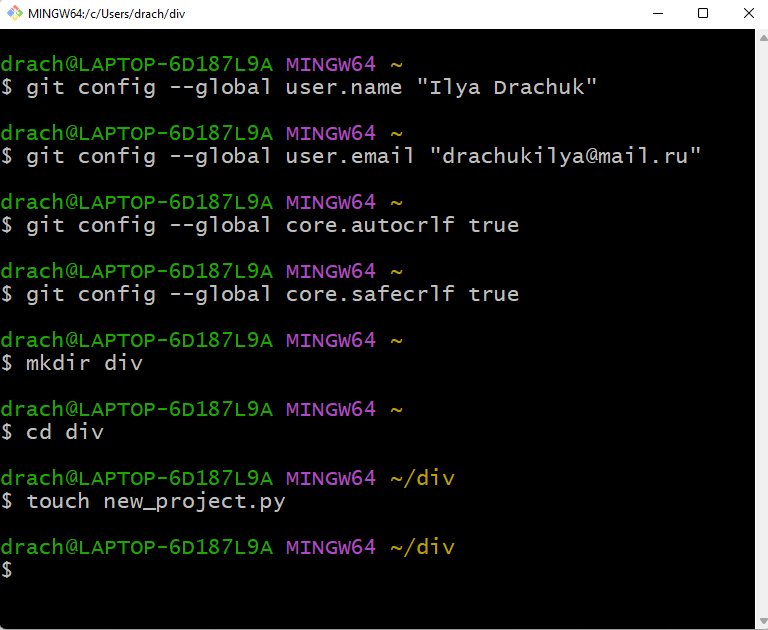
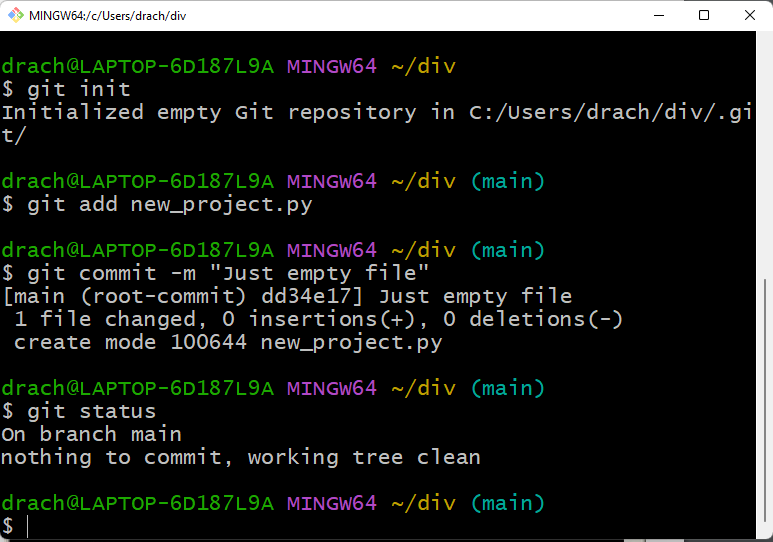
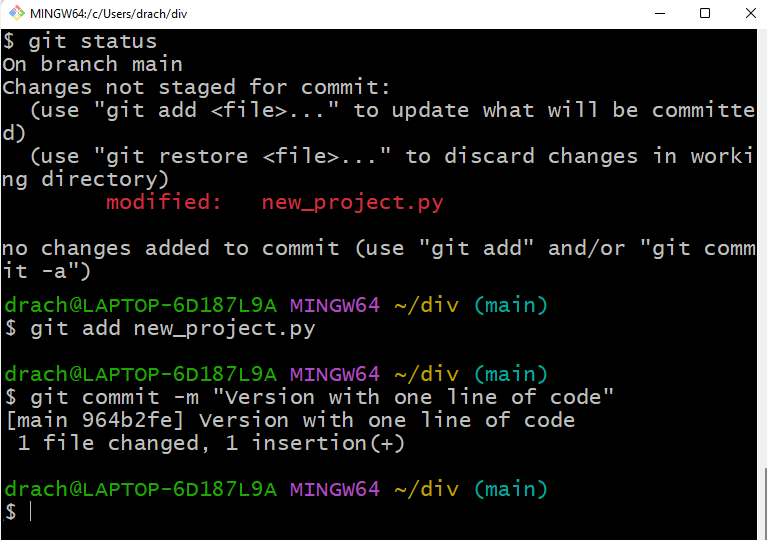
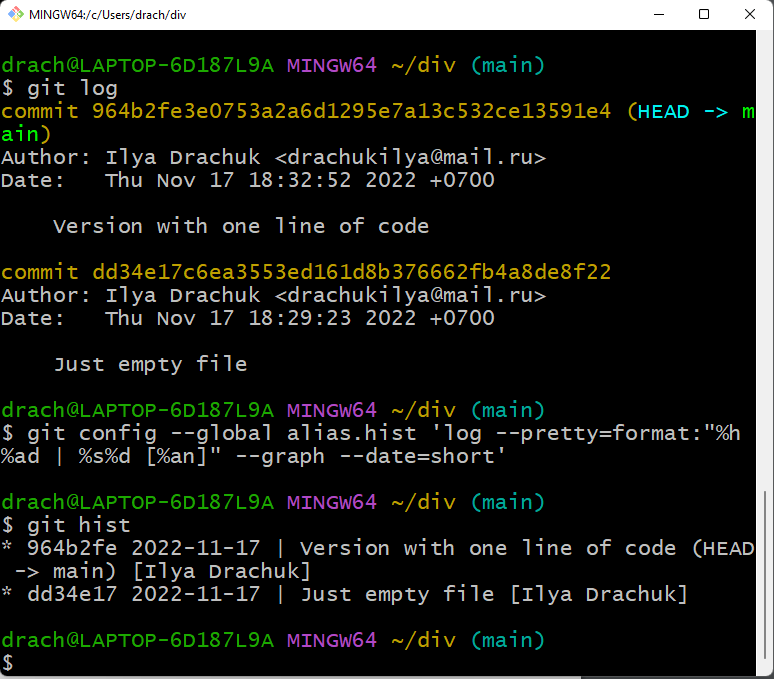
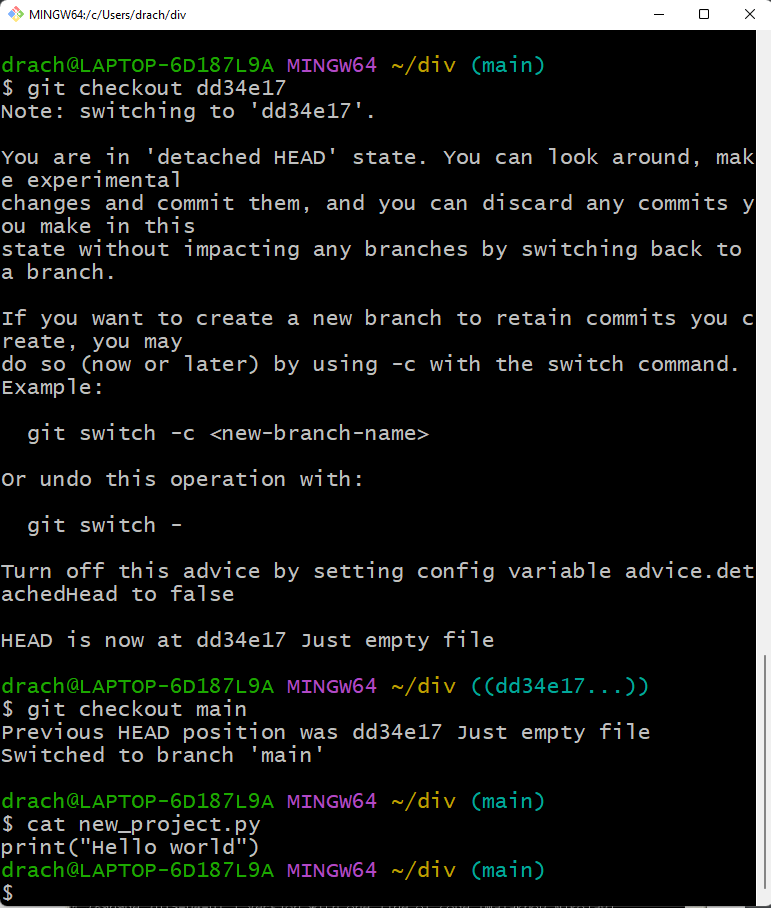
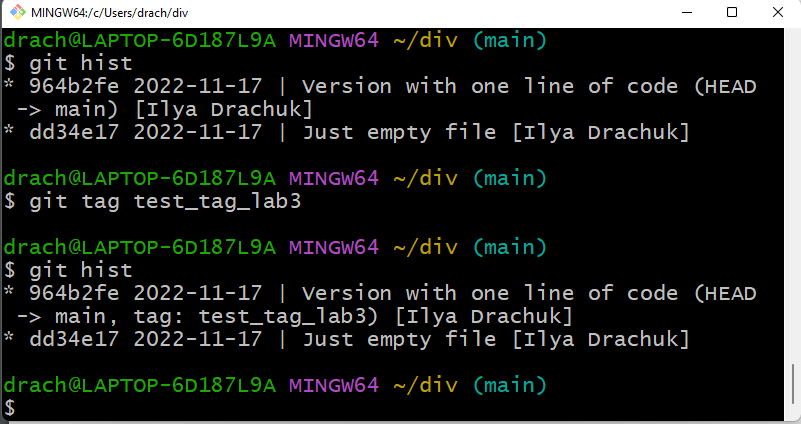
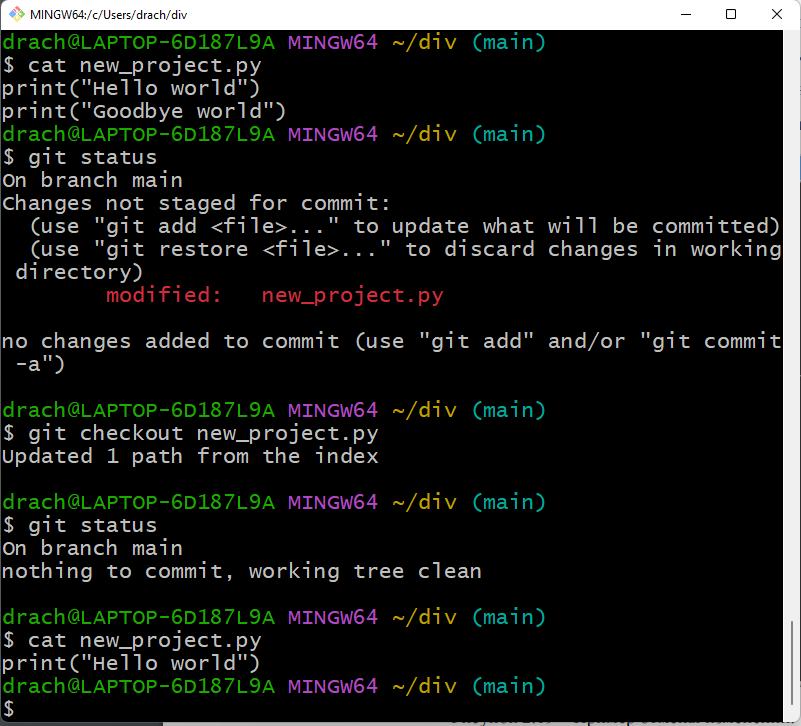
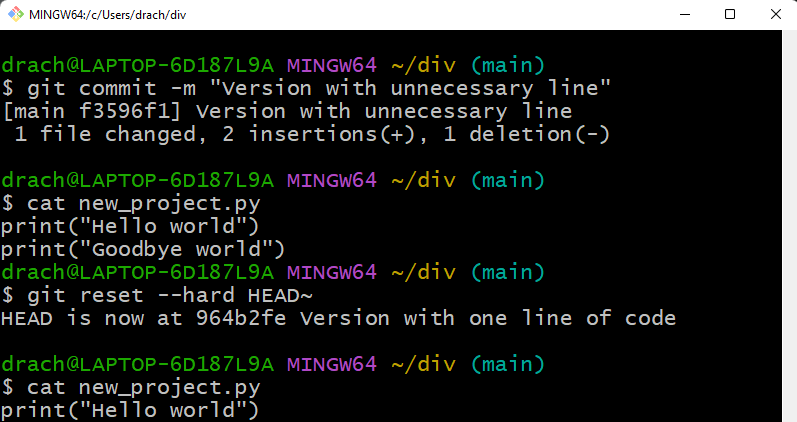
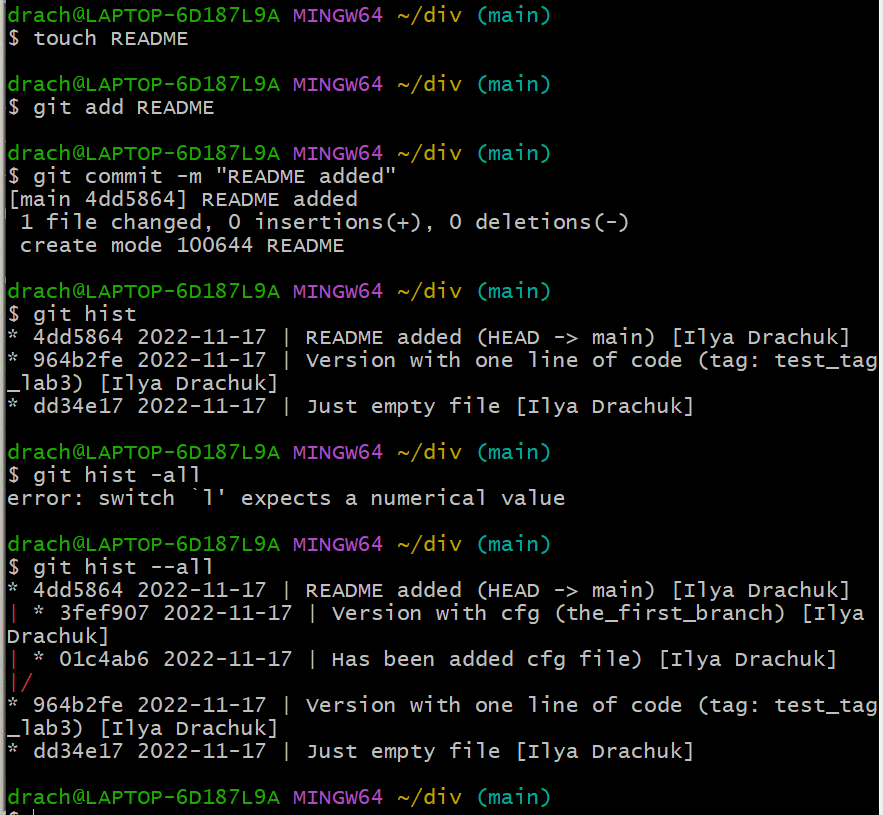
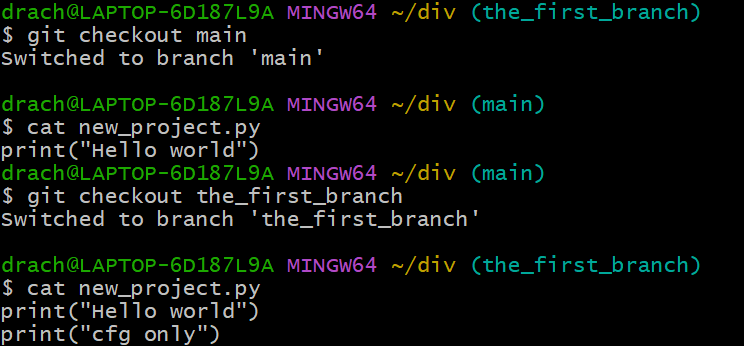
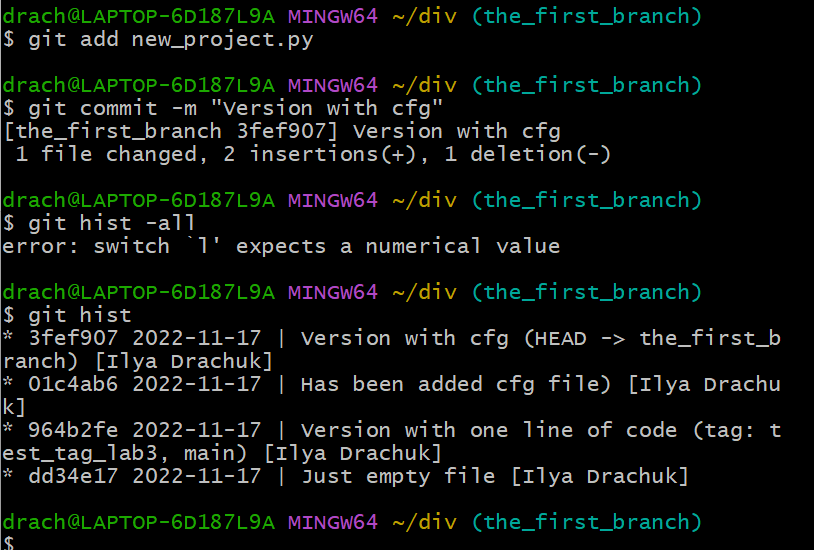
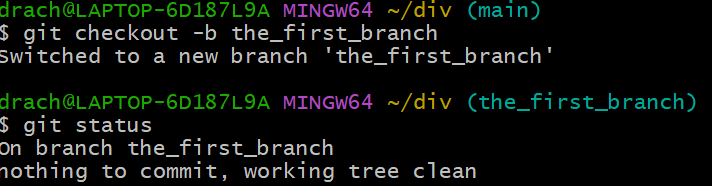
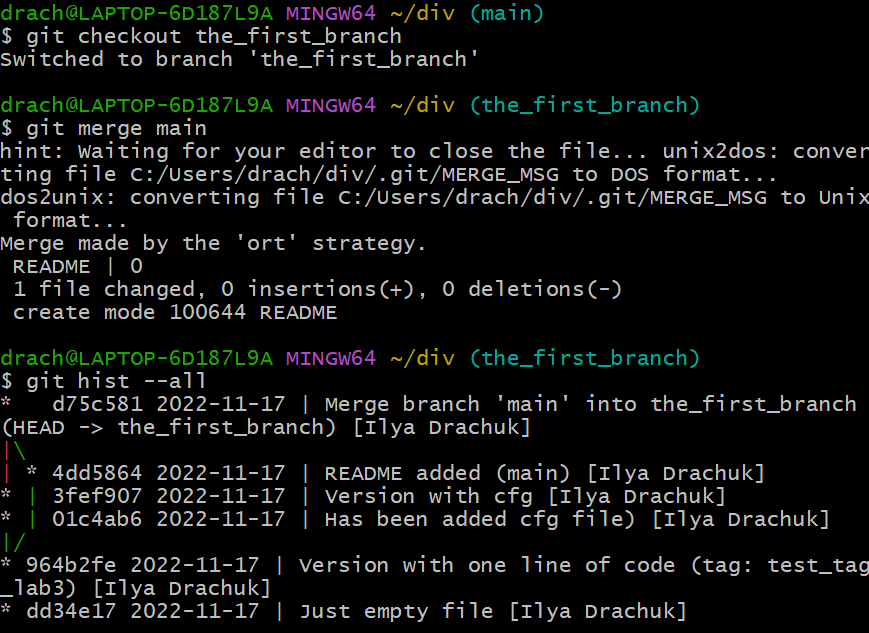
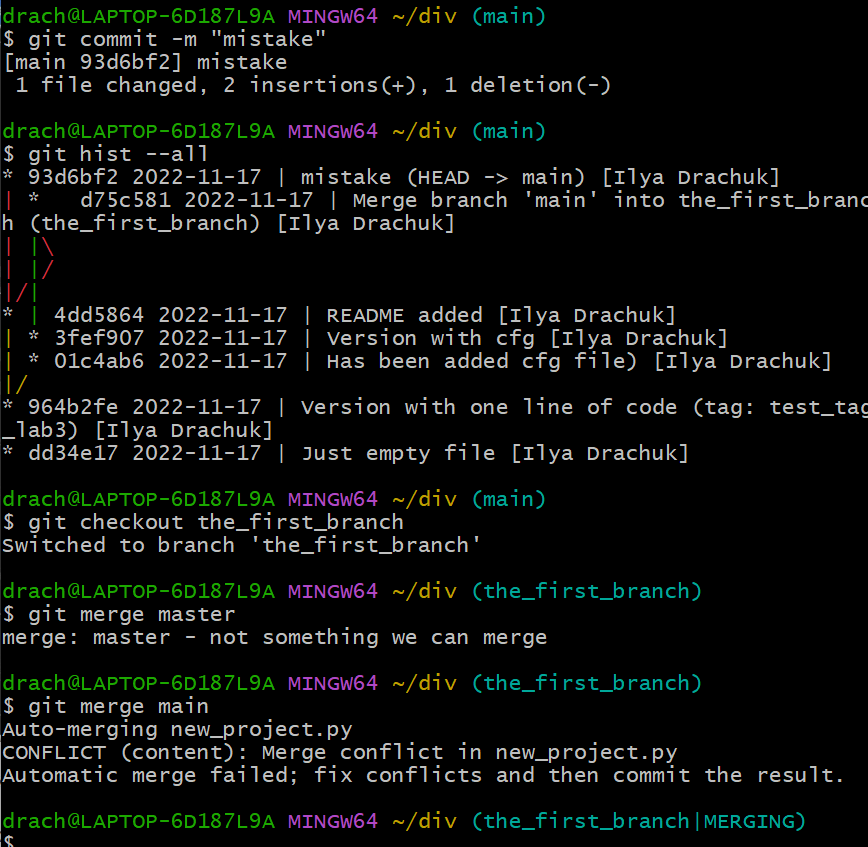
**Отчёт по лабораторной работе №3**

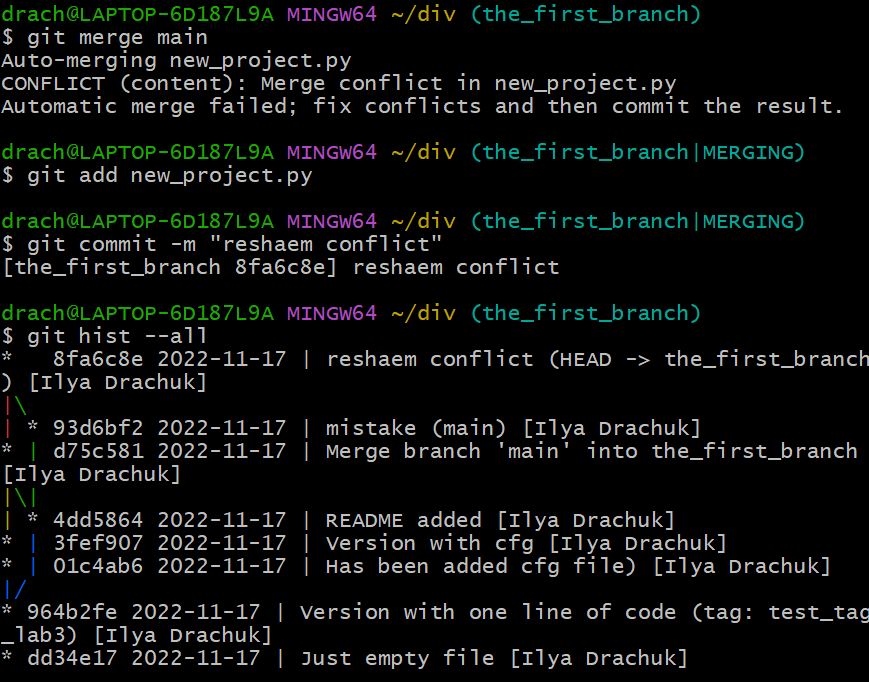
**Драчук Илья БПИ22-01 Вариант-7**

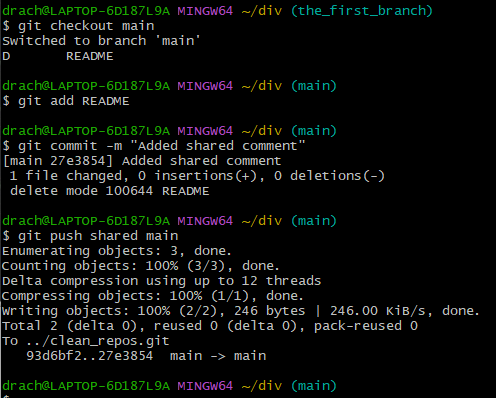
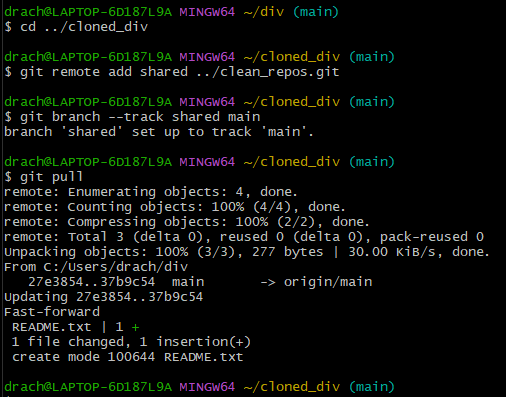
Основное задание:

* Первоначальная настройка, создание директории и файла
* Создание репозитория, добавление туда файла, проверка состояния репозитория
* Проверка состояния после изменения файла, коммит измененного файла
* Просмотр изменений через стандартную команду и через созданный пользователем алиас
* Возвращаемся к предыдущей версии файла, а затем снова к новейшей
* Добавим тэг одному из коммитов
* Убираем ненужную строку (коммита не было), откатываясь до последнего коммита
* Пример отката назад после случайного коммита ненужной версии (при использовании git reset –hard HEAD~ он также удалится из истории)
* Создаём новую ветку, работаем с ней\

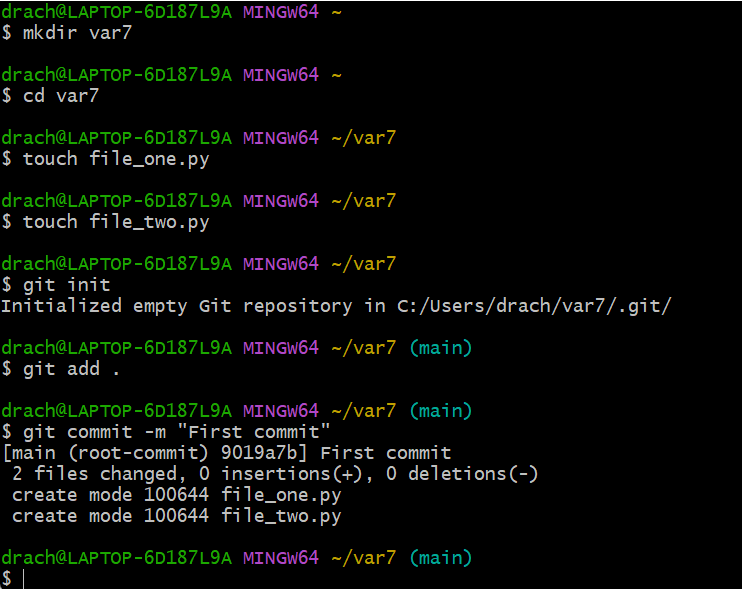
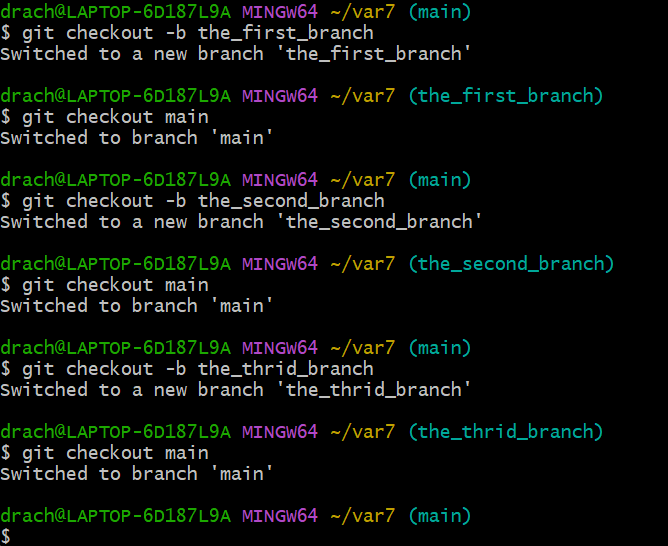
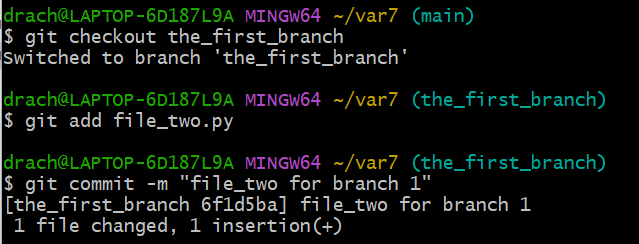
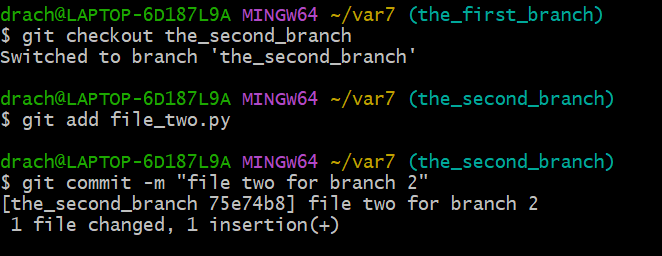
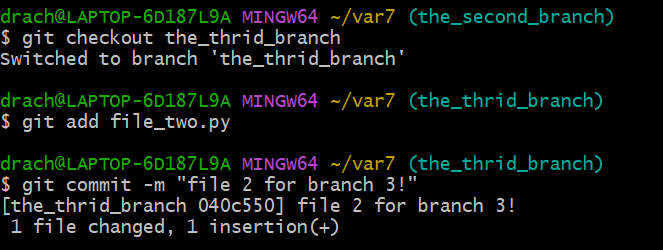
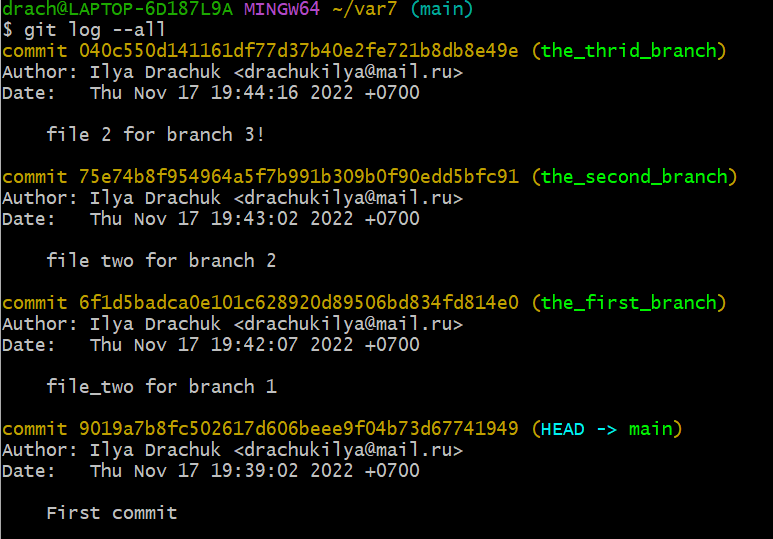
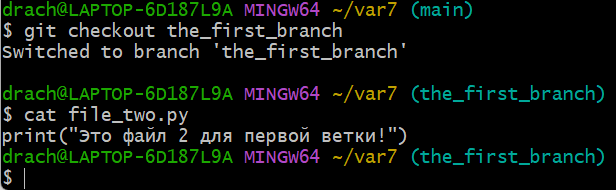
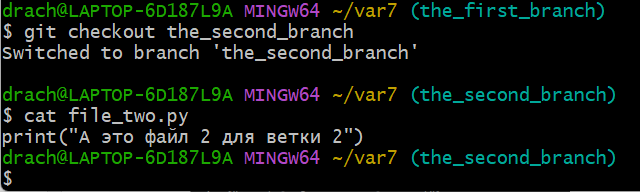
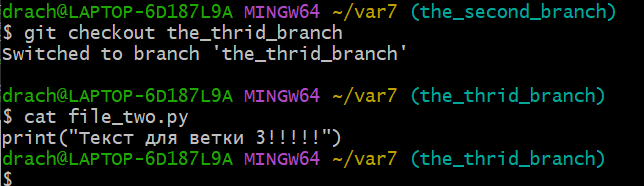
Слияние веток

Создаём конфликт

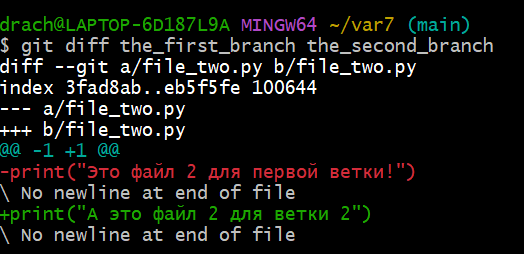
Решаем его

* Создание чистого репозитория, работа с ним
* Создание удалённого репозитория, работа с ним

Задание Варианта 7: Использовать в работе 2 разных файла, для одного из них использовать 3 различные ветки.

* Создадим новую директорию, в ней создадим два файла, инициализируем репозиторий, сделаем коммит 
* Создадим ветки, по которым будем распределять file\_two.py 
* Переходим на каждую ветку и добавляем на каждую из них разные версии файла file\_two.py   
* Посмотрим лог 
* А теперь убедимся, что содержание file\_two.py на разных ветках - разное   

Ответы на контрольные вопросы:

1. GIT – Система контроля версий, которая позволяет контролировать изменения в файлах, откатывать эти изменения, регистрировать, кто и когда внёс изменения и т.д.
2. Отслеживает изменения в файлах, предоставляет возможность создания новых веток и слияния существующих, проводит контроль доступа пользователей к проекту, определяет, кто, когда и какие внёс изменения
3. Коммит – операция отправки изменений в систему контроля версий, он выполняется, когда пользователь хочет добавить текущую обновленную версию файла в систему контроля версий.
4. С помощью команды git log или заданных вручную алиасов с аналогичным функционалом
5. С помощью команды git init
6. Архитектура деревьев (ветвления)
7. git checkout -b “название ветки”, для соединения git merge “ветка, с которой нужно соединиться”
8. stage это место, куда отправляется необходимый набор файлов, в которые были внесены изменения. Они находятся там до коммита.
9. Например, с помощью команды git diff 

Вывод: Система контроля версий является очень удобным инструментов как для одного разработчика, так и для команды из нескольких людей. Она позволяет более гибко работать над любым проектом при помощи системы ветвления, которые в последствии можно соединить.