**Основы работы с ветвями в GIT. Основы слияния. Разрешение конфликтов.**

Часть 1. Создание ветвей и слияние.

1. Создайте новую ветвь greet\_name (где name – Ваша фамилия), используя команду

git checkout -b <branchname>



1. Добавьте новый каталог (пример lib\_name) и файл *greeter\_name.rb* c содержимым:

class Greeter

def initialize(who)

@who = who

end

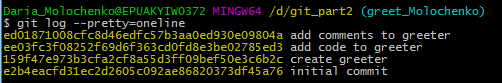
def greet

"Hello, #{@who}"

end

end

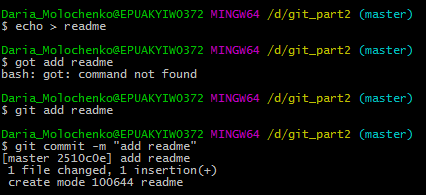
1. Сделайте 3 commits в ветку greet\_name

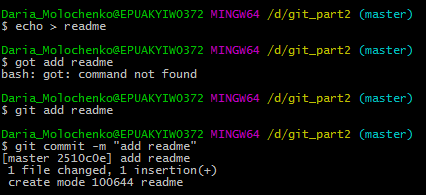


1. Продемонстрируйте переключение на ветку master.

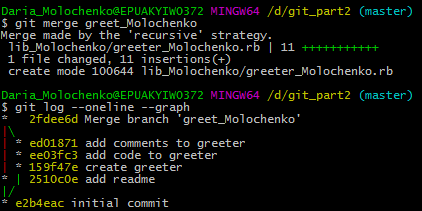


1. Создайте файл README и произведите commit в master.

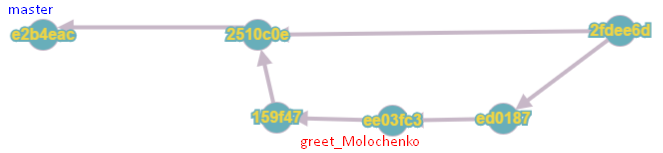




1. Произведите слияние ветвей greet и master – продемонстрируйте историю commits.

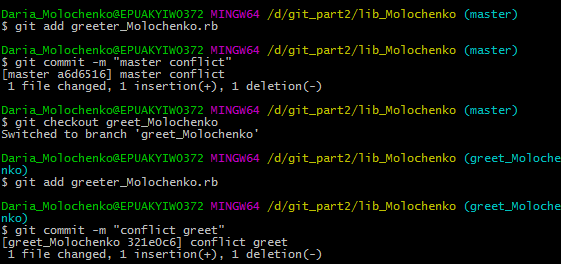


1. Изобразите в удобном для Вас графическом редакторе граф коммитов, где вершинами графа будут коды коммитов.

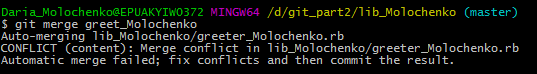


Часть 2. Смоделируйте искусственно конфликт, при слиянии описанном в предыдущей части.

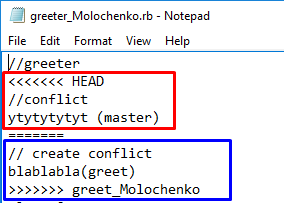
***Изменяем одни и те же строчки в файле greeter\_name.rb на ветках master и greet\_molochenko. Фиксируем изменения.***



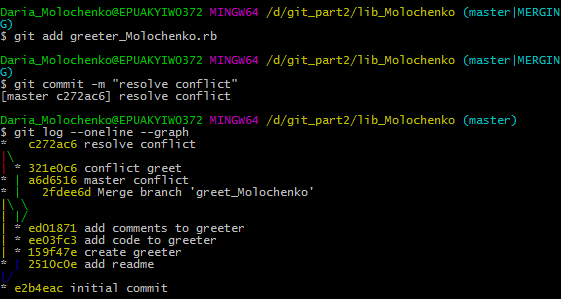
***Пытаемся сделать merge:***



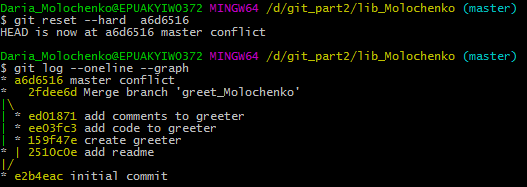
***Получаем конфликт. Решаем его в ручную:***



***Фиксируем изменения:***

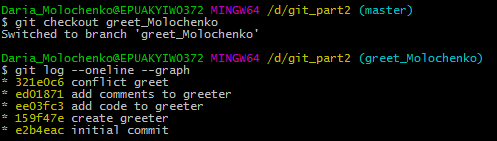


Часть 3. Используя команду reset верните указатель на состояние ветвей до их слияния.



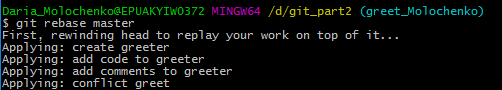


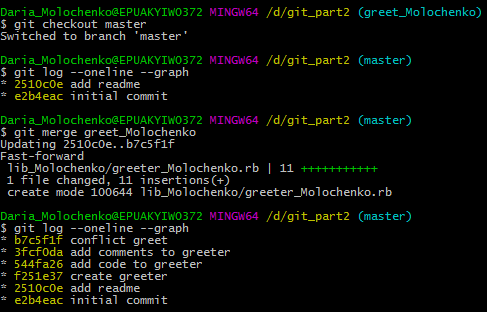


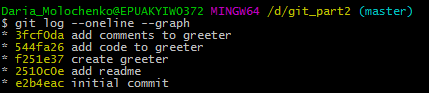


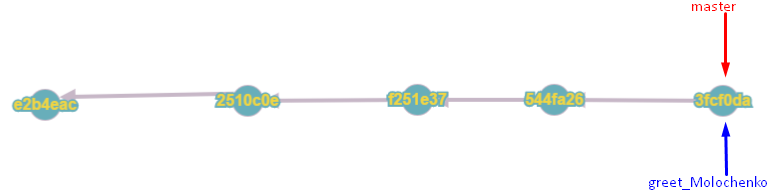
Часть 4.

Проделайте шаги 3-5 в части 1. Выполните слияние изменений командой rebase. Изобразите в удобном для Вас графическом редакторе граф коммитов, где вершинами графа будут коды коммитов.







Объясните разницу между merge и rebase. На примере Ваших графов.

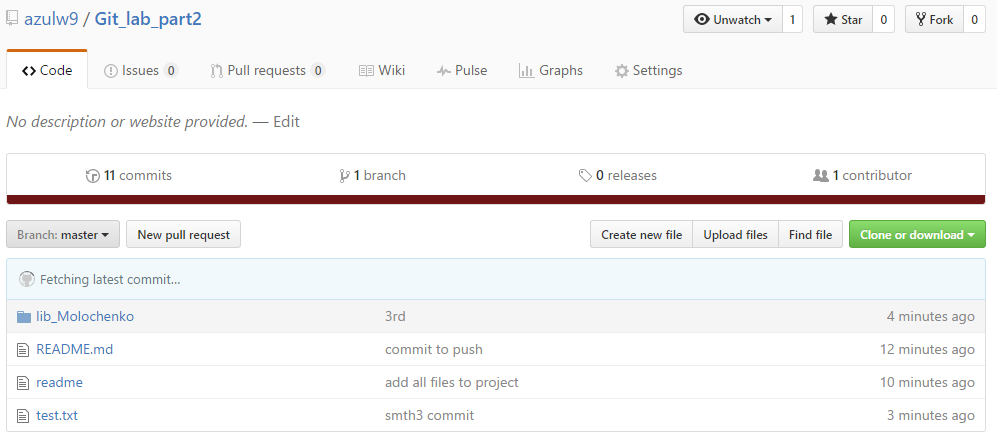
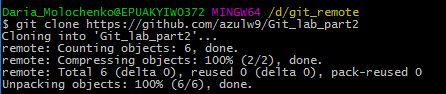
***В результате мы получаем одно и тоже состояние репозитория как после merge, так и после rebase. Но rebase позволяет сохранить историю коммитов с разных веток и при этом эта история будет линейной.А лишний ветки можна удалить, пока они не нужны. Это удобней, чем куча разветвлений, в которых можно запутаться.***

***Если использовать merge, то мы или будем накапливать ветки и тем самым получать сложную разветвленную структуру или же вынуждены будем удалять ветки, для сохранения линейности истории, и тем самым удаляя историю коммитов, которые могут пригодиться в будущем.***

Для начала работы создайте/зарегистрируйтесь (в случае отсутствия на текущий момент) на удаленном сервисе (типа *github*) *,* т.е. инициализируйте удаленный репозиторий.

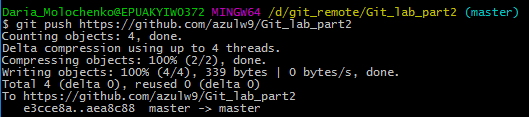
Часть 5.

1. Разбейтесь на пары.
2. Выберите в качестве общего репозитория один из двух удаленных.

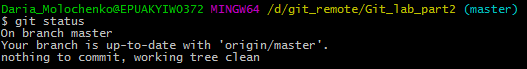


1. Student1 заливает все содержимое своих папок в удаленный репозиторий.

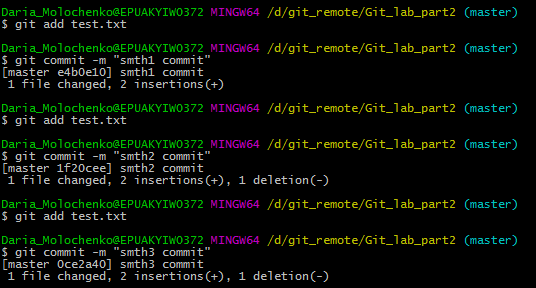
Student 2 «забирает» проект к себе в локальный репозиторий.



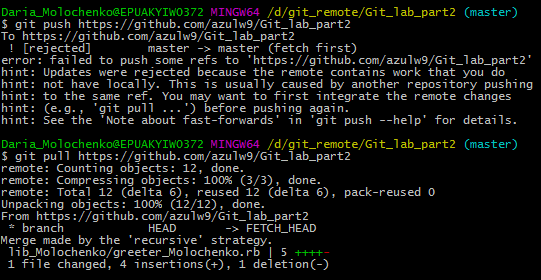
1. Выполните команду git status.



1. Проведите каждый по три коммита в локальный репозиторий и «залейте изменения» в удаленный.



***Пытаемся залить изменения:***



***Но, другой учасник уже залил свои и потому состояние моего локального репозитория устарело,а на удаленном есть новые изменения. Потому сначала делаем git pull, а потом git push. Конфликтов нет, потому при pull происходит automerge.***

***По git status видим, что все хорошо, все изменения были внесены в удаленный репозиторий и состояния локального и удаленного репозитория одинаковые.***

