Błażej Naziemiec
grupa 12
Raport rozwiązań zadań z gita
https://gitexercises.fracz.com/committer/8f336cf6ab9fcb8a1860f3479373747237f98c79

1. master

Aby zrobić to zadanie wystarczyło wpisać $git\ start\ master$

2. commit-one-file

Najpierw użyłem komendy $git\ add\ A.txt$ aby dodać do commita plik A.txt. Następnie aby utworzyć ten commit wpisałem
$git\ commit\ -m\ "Commit\ A.txt\ file"$

3. commit-one-file-staged

Na początku aby sprawdzić, jakie pliki są dodane wpisałem $git\ status$, a następnie aby usunąć z commita plik A.txt wpisałem
$git\ restore\ --staged\ A.txt$. Dla upewnienia się, czy wszystko jest dobrze ustawione ponownie wpisałem $git\ status$, a potem
$git\ commit\ -m\ "Commit\ B.txt\ file"$

4. ignore-them

Najpierw w pliku $.gitignore$ ręcznie dodałem $*.o$, $*.exe$, $*.jar$ oraz $libraries/$. Potem w git bash za pomocą komendy $git\ add\ .gitignore$ dodałem ten plik do commita, a później z użyciem $git\ commit\ -m\ "gitignore"$ stworzyłem ten commit

5. chase-branch

Aby zrobić to zadanie trzeba było zrobić merge brancha $chase-branch$ z $escaped$ wpisałem $git\ merge\ escaped$

6. merge-conflict

Na początek zrobiłem merge aktualnego brancha za pomocą
$git\ merge\ another-piece-of-work$, potem rozwiązuje konflikt, żeby potem za pomocą komend

$git\ add\ equation.txt$
$git\ commit\ -m\ "conflict\ fixed"$

dodałem commit z rozwiązanym konfliktem

7. save-your-work

Aby zapisać aktualny stan plików użyłem $git\ stash$. Następnie poprawiłem buga i za pomocą $git\ commit\ -am\ "fix\ bug"$ utworzyłem commita z naprawionym kodem. Potem zwróciłem za pomocą $git\ stash\ pop$ wszystkie

pliki, których stan zapisałem, dodałem wymaganą linijkę do pliku i dodałem nowy commit

8. change-branch-history

Aby zmienić historię brancha należy wpisać komendę
$git\ rebase\ hot\ -\ bugfix$

9. remove-ignored

Aby usunąć plik, który znajduje się w $.gitignore$, a został dodany do git przed utworzeniem tego pliku należy wpisać komendy $git\ rm\ ignored.txt$ oraz $git\ commit\ -\ am$ "$remove\ file\ created\ before\ .gitignore$" i od tego momentu git usunie ten plik oraz będzie go ignorować przy przyszłych commitach

10. case-sensitive-filename

Aby zmienić nazwę pliku należy wpisać komendy $git\ mv\ File.txt\ file.txt$ a następnie należy utworzyć commmit $git\ commit\ -\ am$ "$renamed\ file$"

11. fix-typo

Na początku poprawiłem literówkę w pliku, dodałem ten plik w git, a następnie użyłem $git\ commit\ -\ amend$. Dzięki tej komendzie byłem w stanie zmienić nazwę wcześniejszego commita

12. forge-date

Aby zmienić datę commita wpisałem komendę
$git\ commit\ --\ amend\ --\ no\ -\ edit\ --\ date = "1987\ -\ 08\ -\ 03"$

13. fix-old-typo

Najpierw użyłem komendy $git\ rebase\ -\ i\ HEAD{\sim}2$ aby móc zmienić typ commita 2 wstecz z $pick$ na $edit$. Następnie poprawiłem literówkę, użyłem ponownie $git\ commit\ -\ amend$ żeby edytować ostatni commit. Poprawiłem tam literówkę, a na koniec aby wrócić do najnowszego commita wpisałem $git\ rebase\ -\ continue$, $git\ add$. oraz $git\ rebase\ -\ continue$

14. commit-lost

Aby zobaczyć wszystkie poprzednie commity wpisuje $git\ reflog$ i kopiuje hash szukanego commita. Następnie aby cofnąć zmianę w szukanym commit wpisuje $git\ reset\ -\ hard\ hash$

15. split-commit

Aby usunąć zmiany w poprzednim commit wpisuję
$git\ reset\ -\ mixed\ HEAD{\sim}1$, a następnie w dwóch commitach wysyłam oba pliki

16. too-many-commits

Aby zobaczyć dwa ostatnie commity wpisuje $git\ log\ -2$. Następnie wpisuje $git\ reabse\ -interactive$ i zmieniłem status późniejszego commita z $pick$ na $squash$

17. executable

Aby dodać możliwość, żeby plik był wykonywalny należy wpisać w konsoli komendę $git\ update-index\ –chmod=+\ x\ script.sh$. Następnie dodaje nowy commit zawierający tą zmianę

18. commit-parts

Aby dodać tylko część pliku do najnowszego commita wpisuje $git\ add\ -p$. Ze wszystkich opcji, które git wypisał wybrałem opcję s. Następnie w zależności, czy linijka zawierała informację na temat task 1 wpisywałem y lub n. Następnie dodaję oba commity

19. pick-your-feature

Aby połączyć dwa branche użyłem komendy $git\ cherry-pick$. Będąc na branchu $pick-your-feature$ wpisałem $git\ cherry-pick\ feature-a$, potem $git\ cherry-pick\ feature-b$, a na koniec $git\ cherry-pick\ feature-c$ i rozwiązujemy powstały konflikt. Na sam koniec wpisujemy $git\ cherry-pick\ -continue$

20. rebase-complex

Aby połączyć dwa branche pomijając konkretny branch wpisujemy komendę $git\ rebase\ –onto\ your-master\ issue-555\ rebase-complex$ (do jakiego brancha chcemy dołączyć, bez jakiego brancha, który branch chcemy dołączyć)

21. invalid-order

Aby zamienić kolejnością commity wpisuję $git\ rebase\ -i\ HEAD\sim 2$ i ręcznie zmieniam kolejność obu commitów