executable

```
chmod +x script.sh
git commit -am "Make executable"
git verify
```

forge-date

```
git commit --amend --no-edit --date "$(git log | grep Date | head -n 1 | sed 's/Date:\s*//;s/2022/1987/')" # Меняем год в дате коммита на 1987
git verify # Проверка задания
```

commit-lost

```
git checkout HEAD@{1}
git branch -D commit-lost
git checkout -b commit-lost
git verify
```

ignore-them

```
touch .gitignore
echo "*.exe" >> .gitignore # Игнорировать *.exe
echo "*.o" >> .gitignore # Игнорировать *.o
echo "*.jar" >> .gitignore # Игнорировать *.jar
echo "libraries/*" >> .gitignore
git add -A # Застейджить изменения (file.txt и .gitignore)
git commit -m ignore-them # Закоммитить изменения
git verify # Проверка задания
```

commit-parts

```
git add -p # В интерактивном режиме добавляем изменения относящиеся к Task 1
git commit -m "Task 1"
git commit -am "Task 2"
git verify
```

fix-old-typo

```
git reset --soft HEAD~1 # Нужно изменить предыдущий коммит, отменяем последний
git stash
sed -i 's/wordl/world/g' file.txt
git commit --amend --all --no-edit -m "Add Hello world"
git stash pop
vim file.txt # Разбираемся с конфликтами
git commit -am "Further work on Hello world"
git verify # Проверка задания
```

invalid-order

```
git reset --soft HEAD~1
git stash
git reset --soft HEAD~1
git stash
git stash apply stash@{1}
git commit -am "This should be the first commit"
git stash pop
git commit -am "This should be the second commit"
git verify
```

merge-conflict

```
git merge another-piece-of-work # Начала слияния
# Вывод показывает конфликт в файле equation.txt
echo "2 + 3 = 5" > equation.txt # Файл содержал или "2 + ? = 5" или "? + 3 = 5", мы хотим получить из этого "2 + 3 = 5"
git add -A # Застейджить изменения
git commit -m merge-conflict # Закоммитить изменения
git verify # Проверка задания
```

save-your-work

```
git stash # Сохраняем незакоммиченные изменения
sed -i "/THIS IS A BUG/d" bug.txt # Удалить строку с багом
git add -A # Застейджить изменения
git commit -m save-your-work # Закоммитить изменения
git stash pop # Возвращаем незакоммиченные изменения
echo "Finally, finished it!" >> bug.txt # Завершаем работу
git add -A # Застейджить изменения
git commit -m save-your-work-2 # Закоммитить изменения
git verify # Проверка задания
```

search-improved

```
for COMMIT in $(git rev-list HEAD 1.0); do git checkout $COMMIT; echo $COMMIT; ./faulty-check cool-cheatsheet.txt && echo OK || echo BAD; done
# Команда выведет для всех "плохих" коммитов BAD, а для "хороших" - OK. Из её вывода понимаем, что последний "плохой" коммит - 2130ffa401b6b6d7f66e1c295a1b0769e1a3414b
git push origin 2130ffa401b6b6d7f66e1c295a1b0769e1a3414b:search-improved
```

too-many-commits

```
git reset --soft HEAD~2
git commit -am "Add file.txt"
git verify
```

change-branch-history

```
git rebase hot-bugfix # Переместить ветку на hot-bugfix
git verify # Проверка задания
```

commit-one-file-staged

```
git reset B.txt # Убрать файл B.txt из списка застейдженных
git commit -m commit-one-file-staged # Закоммитить изменения
git verify # Проверка задания
```