Задание

Цель работы: получить навыки по работе с командной строкой и git’ом.

Часть 1. Основные команды Git

1. Установите и настройте клиент git на своей рабочей станции.

2. Создайте локальный репозиторий и добавьте в него несколько файлов.

```
C:\Users\KoV>mkdir my_repo

C:\Users\KoV>cd my_repo

C:\Users\KoV\my_repo>git init
Initialized empty Git repository in C:/Users/KoV/my_repo/.git/
```

```
C:\Users\KoV\my_repo>echo "Hello, world" > file1.txt

C:\Users\KoV\my_repo>echo "One more file" > file2.txt

C:\Users\KoV\my_repo>
```

3. Внесите изменения в один из файлов.

```
C:\Users\KoV\my_repo>echo "Добавленная строка" >> file1.txt
```

4. Проиндексируйте изменения и проверьте состояние.

```
C:\Users\KoV\my_repo>git add file1.txt

C:\Users\KoV\my_repo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2.txt
```

5. Сделайте коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.

```
C:\Users\KoV\my_repo>git commit -m "Добавленная строка в "file1.txt"
[master (root-commit) d450110] Добавленная строка в file1.txt
 1 file changed, 2 insertions(+)
 create mode 100644 file1.txt
```

6. Измените еще один файл. Добавьте это изменение в индекс git. Измените файл еще раз. Проверьте состояние и произведите коммит проиндексированного изменения. Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды `git status`. Сделайте коммит второго изменения.

```
C:\Users\KoV\my_repo>echo "Изменения в file2.txt" >> file2.txt

C:\Users\KoV\my_repo>git add file2.txt

C:\Users\KoV\my_repo>echo "Еще одно изменение в file2.txt" >> file2.txt

C:\Users\KoV\my_repo>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2.txt


C:\Users\KoV\my_repo>git commit -m "Первон изменение в file2.txt"
[master 37053f8] Первон изменение в file2.txt
 1 file changed, 2 insertions(+)
 create mode 100644 file2.txt

C:\Users\KoV\my_repo>git add file2.txt

C:\Users\KoV\my_repo>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file2.txt


C:\Users\KoV\my_repo>git commit -m "Второе изменение"
[master 0b703ca] Второе изменение
 1 file changed, 1 insertion(+)
```

7. Просмотрите историю коммитов с помощью команды `git log`. Ознакомьтесь с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов.

```
C:\Users\KoV\my_repo>git log
commit 0b703cac8195d4fe5955b4a8db1dced13e652a9c (HEAD -> master)
Author: Alex Kovalev <167441571+KoooV@users.noreply.github.com>
Date:   Fri Feb 21 13:57:49 2025 +0300

    Второе изменение

commit 37053f8002de7f79d35fadff7f5504e876e8f974
Author: Alex Kovalev <167441571+KoooV@users.noreply.github.com>
Date:   Fri Feb 21 13:56:51 2025 +0300

    Первон изменение в file2.txt

commit d450110b720d306e5b7c5fd9cf232bb5ad92a554
Author: Alex Kovalev <167441571+KoooV@users.noreply.github.com>
Date:   Fri Feb 21 13:54:33 2025 +0300

    Добавленная строка в file1.txt
```

```
C:\Users\KoV\my_repo>git log --oneline
0b703ca (HEAD -> master) Второе изменение
37053f8 Первон изменение в file2.txt
d450110 Добавленная строка в file1.txt
```

```
C:\Users\KoV\my_repo>git log --graph --decorate --all
* commit 0b703cac8195d4fe5955b4a8db1dced13e652a9c (HEAD -> master)
| Author: Alex Kovalev <167441571+KooоV@users.noreply.github.com>
| Date:   Fri Feb 21 13:57:49 2025 +0300
|
|     Второе изменение
|
* commit 37053f8002de7f79d35fadff7f5504e876e8f974
| Author: Alex Kovalev <167441571+KooоV@users.noreply.github.com>
| Date:   Fri Feb 21 13:56:51 2025 +0300
|
|     Первон изменение в file2.txt
|
* commit d450110b720d306e5b7c5fd9cf232bb5ad92a554
  Author: Alex Kovalev <167441571+KooоV@users.noreply.github.com>
  Date:   Fri Feb 21 13:54:33 2025 +0300

      Добавленная строка в file1.txt
```

8. Верните рабочий каталог к одному из предыдущих состояний.

```
C:\Users\KoV\my_repo>git checkout HEAD~1
Note: switching to 'HEAD~1'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 37053f8 Первон изменение в file2.txt
```

9. Изучите, как создавать теги для коммитов для использования в будущем.

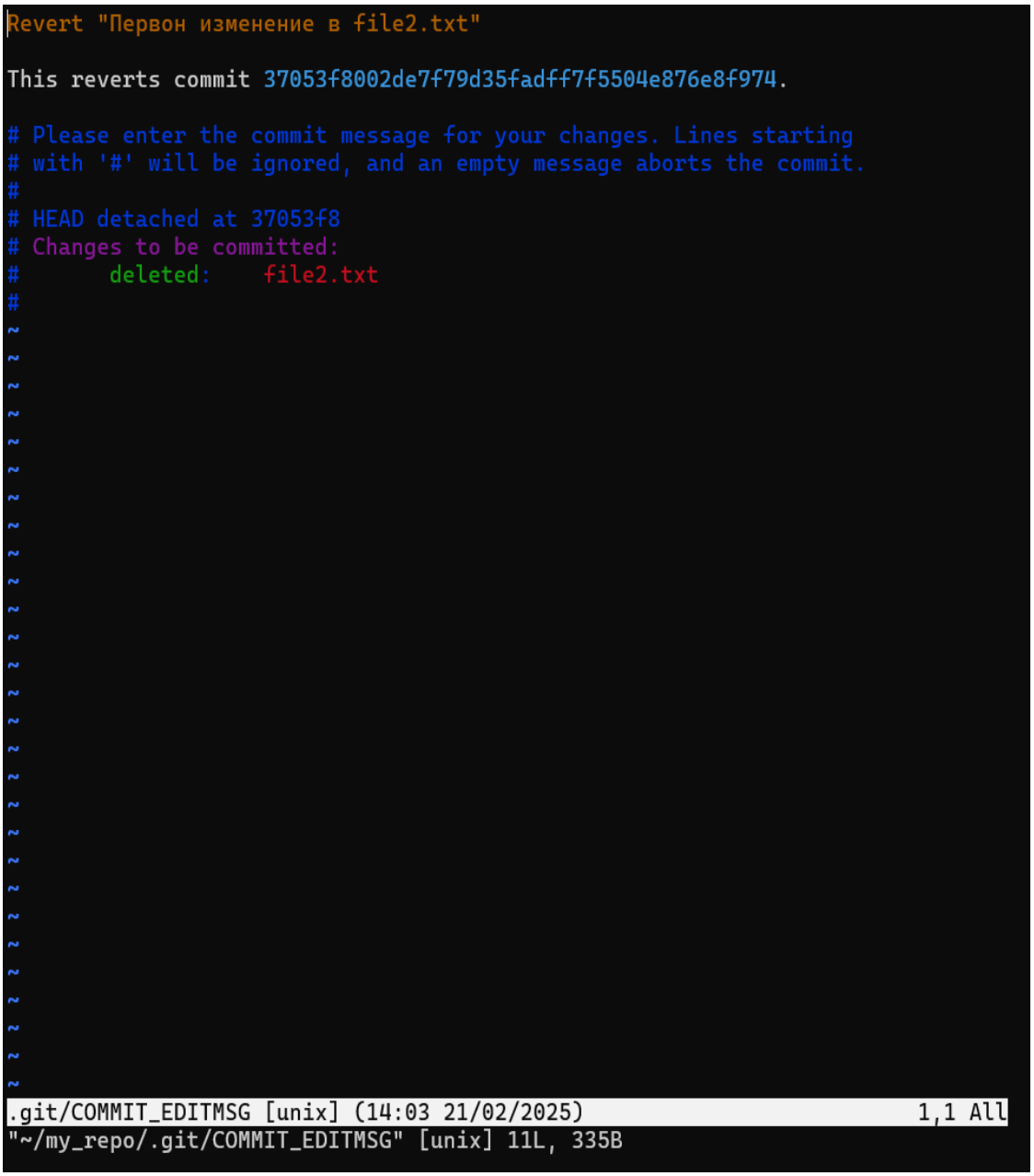
```
C:\Users\KoV\my_repo>git tag -a v1.0 -m "Version 1.0"
```

10. Отмените некоторые изменения в рабочем каталоге (до и после индексирования).

```
C:\Users\KoV\my_repo>git checkout -- file2.txt

C:\Users\KoV\my_repo>git reset HEAD file2.txt
```

11. Отмените один из коммитов в локальном репозитории.



Часть 2. Системы управления репозиториями

1. Создайте аккаунт на GitHub (у кого нет),

   https://github.com/KoooV

2. Создайте репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта,

```
C:\Users\KoV\study\gitTest>git init
Initialized empty Git repository in C:/Users/KoV/study/gitTest/.git/
```

3. Создайте несколько файлов на локальной машине при помощи консоли,

```
C:\Users\KoV\study\gitTest>echo "description" > README.md

C:\Users\KoV\study\gitTest>echo "source code" > Main.java
```

4. Создайте SSH-ключ для авторизации,

```
C:\Users\KoV\study\gitTest>ssh-keygen -t ed25519 -C "alexkovalev13681@gmail.
com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\KoV/.ssh/id_ed25519):
Created directory 'C:\\Users\\KoV/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\KoV/.ssh/id_ed25519
Your public key has been saved in C:\Users\KoV/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ijIxPtOtVm/SUetZNl9Cfu5IHyxgma7BPUPRbR9vVrk alexkovalev13681@gmail.co
m
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|            . . .|
|           . . =.|
|          . + o *|
|  o     S. B o E=|
| . + o..o * = +oo|
|  * o.oo = O +.B |
|   =... + = o.ooo|
|   ..  o .    ..o|
+----[SHA256]-----+
```

5. Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли,

```
C:\Users\KoV\study\gitTest>git remote add origin https://github.com/KoooV/TR
PP.git
```

```
C:\Users\KoV\study\gitTest>git add .

C:\Users\KoV\study\gitTest>git commit -m "initial commit"
[master (root-commit) d30142b] initial commit
 2 files changed, 2 insertions(+)
 create mode 100644 Main.java
 create mode 100644 README.md

C:\Users\KoV\study\gitTest>git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 310 bytes | 155.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/KoooV/TRPP.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

6. Создайте новую ветку в репозитории с помощью команды, произведите в ней

какие-нибудь изменения, а после слейте с веткой master,

```
C:\Users\KoV\study\gitTest>git chechout -b secondBranch
git: 'chechout' is not a git command. See 'git --help'.

The most similar command is
        checkout

C:\Users\KoV\study\gitTest>git checkout -b secondBranch
Switched to a new branch 'secondBranch'

C:\Users\KoV\study\gitTest>echo "New feature" >> README.md

C:\Users\KoV\study\gitTest>git add README.md

C:\Users\KoV\study\gitTest>git commit -m "Add new feature"
[secondBranch 971363b] Add new feature
 1 file changed, 1 insertion(+)

C:\Users\KoV\study\gitTest>git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

C:\Users\KoV\study\gitTest>git merge secondBranch
Updating d30142b..971363b
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)

C:\Users\KoV\study\gitTest>git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 163.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/KoooV/TRPP.git
   d30142b..971363b  master -> master
```

7. Выполните цепочку действий в репозитории, согласно вариантам.

Обоснование: V=Ост((N-1)/M)+1

N = 11(номер в списке), M = 11(количество тем)

V = Ост((11-1)/11)+1 = 10 + 1 = 11

Ост(10/11) = 10

Вариант 11.

1) Клонируйте непустой удаленный репозиторий на локальную машину

```
C:\Users\KoV\study\gitTest>git clone https://github.com/KoooV/softComparison
Cloning into 'softComparison'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 2), reused 6 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (2/2), done.
```

2) Создайте новую ветку и выведите список всех веток

```
C:\Users\KoV\study\gitTest\softComparison>git checkout -b new-feature
Switched to a new branch 'new-feature'

C:\Users\KoV\study\gitTest\softComparison>git branch -a
  main
* new-feature
  remotes/origin/HEAD -> origin/main
  remotes/origin/main

C:\Users\KoV\study\gitTest\softComparison>
```

3) Произведите 3 коммита в новой ветке в разные файлы

```
C:\Users\KoV\study\gitTest\softComparison>echo "change 1" > file1.txt
```

```
C:\Users\KoV\study\gitTest\softComparison>git add file1.txt

C:\Users\KoV\study\gitTest\softComparison>git commit -m "add file1.txt"
[new-feature 7896013] add file1.txt
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

C:\Users\KoV\study\gitTest\softComparison>echo "change 2" > file2.txt

C:\Users\KoV\study\gitTest\softComparison>git add file2.txt

C:\Users\KoV\study\gitTest\softComparison>git commit -m "add file2.txt"
[new-feature 315dd17] add file2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

C:\Users\KoV\study\gitTest\softComparison>echo "change 3" > file3.txt

C:\Users\KoV\study\gitTest\softComparison>git add file3.txt

C:\Users\KoV\study\gitTest\softComparison>git commit -m "add file3.txt"
[new-feature e66b706] add file3.txt
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
```

4) Выгрузите изменения в удаленный репозиторий

```
C:\Users\KoV\study\gitTest\softComparison>git push -u origin new-feature
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 888 bytes | 148.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'new-feature' on GitHub by visiting:
remote:      https://github.com/KoooV/softComparison/pull/new/new-feature
remote:
To https://github.com/KoooV/softComparison
 * [new branch]      new-feature -> new-feature
branch 'new-feature' set up to track 'origin/new-feature'.
```

5) Добейтесь того, чтобы эти три изменения были представлены одним коммитом

```
pick 7896013 add file1.txt
squash 315dd17 add file2.txt
squash e66b706 add file3.txt

# Rebase 43555eb..e66b706 onto 43555eb (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#         create a merge commit using the original merge commit's
#         message (or the oneline, if no original merge commit was
#         specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                       to this position in the new commits. The <ref> is
#                       updated at the end of the rebase
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
~
~
~
~
~
~
~
~
~
~
~
~
.git/rebase-merge/git-rebase-todo[+] [unix] (16:44 28/02/2025)          34,2 All
-- INSERT --
```

6) Выведите в консоли различия между веткой master и новой веткой

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTest/softComparison (new-feature|REBASE)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTest/softComparison (new-feature|REBASE)
$ git merge new-feature
Updating 43555eb..e66b706
Fast-forward
 file1.txt | 1 +
 file2.txt | 1 +
 file3.txt | 1 +
 3 files changed, 3 insertions(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
 create mode 100644 file3.txt
```

7) Слейте новую ветку с master при помощи merge

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTest/softComparison (new-feature|REBASE)
$ git merge new-feature
Already up to date.
```

Часть 3. Работа с ветвлением и оформление кода

1. Сделайте форк репозитория в соответствии с вашим вариантом

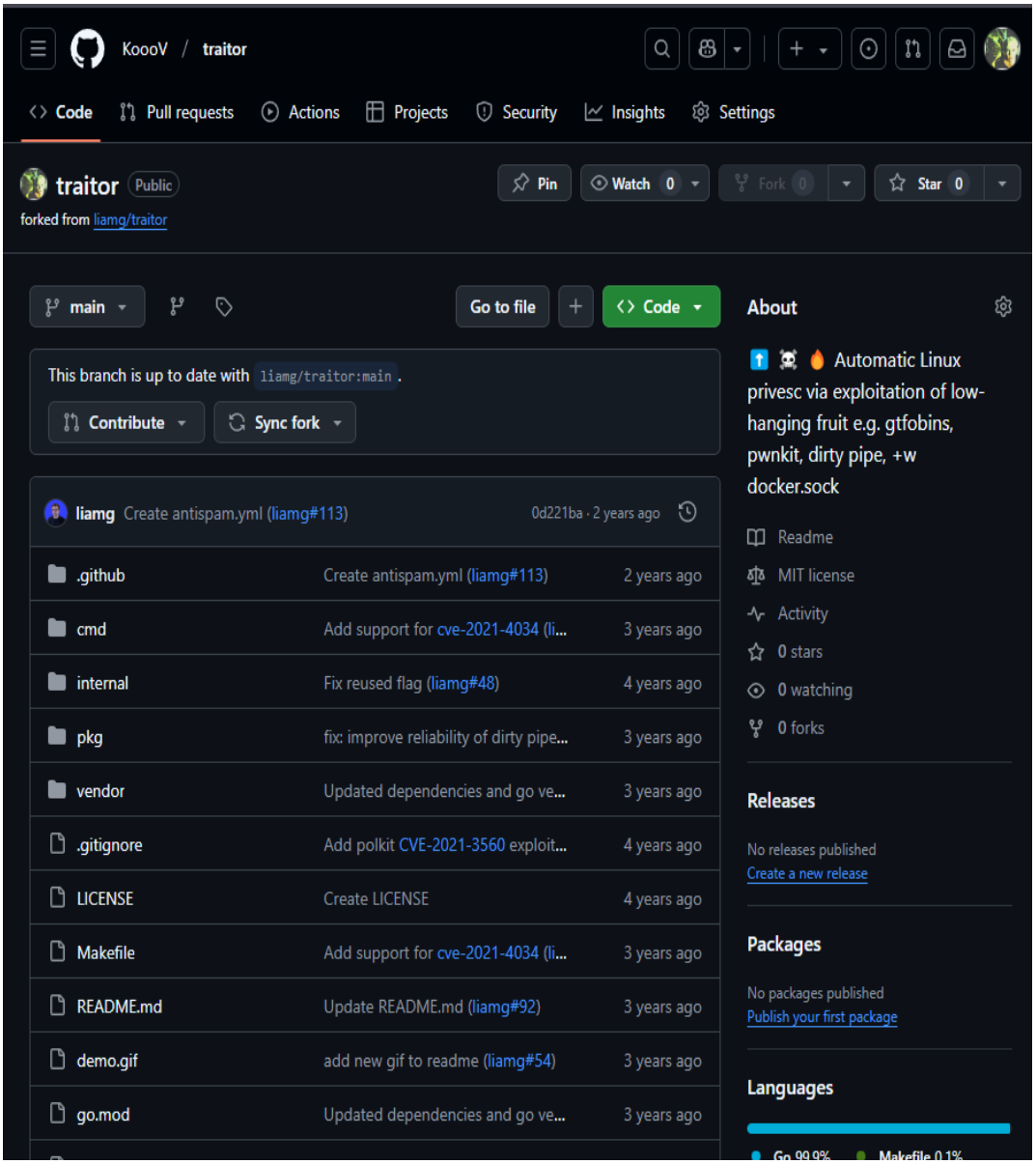
   V=Ост((N-1)/M)+1

   N = 11(номер в списке), M = 11(количество тем)

   V = Ост((11-1)/10)+1 = 0 + 1 = 1 => Вариант 1

   11 -1 = 10

   10 /10 = 1 остаток от деления = 0



2. Склонируйте его на локальную машину

```
oV@DESKTOP-TMH3BFG MINGW64 /
 cd C:/Users/KoV/study/gitTestFork

oV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (spearman)
 git clone https://github.com/KoooV/traitor
loning into 'traitor'...
emote: Enumerating objects: 1354, done.
emote: Counting objects: 100% (185/185), done.
emote: Compressing objects: 100% (61/61), done.
emote: Total 1354 (delta 133), reused 124 (delta 124), pack-reused 1169 (from 1)
eceiving objects: 100% (1354/1354), 2.94 MiB | 702.00 KiB/s, done.
esolving deltas: 100% (704/704), done.

oV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (spearman)
 cd traitor
```

3. Создайте две ветки branch1 и branch2 от последнего коммита в master'e

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (main)
$ git checkout -b branch1
Switched to a new branch 'branch1'

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (branch1)
$ git checkout master
error: pathspec 'master' did not match any file(s) known to git

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (branch1)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (main)
$ git checkout -b branch2
Switched to a new branch 'branch2'
```

4. Проведите по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (branch1)
$ echo "Изменение 1 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (branch1)
$ git add file1.txt
fatal: pathspec 'file1.txt' did not match any files

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork/traitor (branch1)
$ git commit -m "Изменение 1 в branch1"
[branch1 7e5fe16] Изменение 1 в branch1
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
```

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ echo "Изменение 2 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git commit -m "Изменение 2 в branch1"
[branch1 8f72a51] Изменение 2 в branch1
 1 file changed, 1 insertion(+)
 create mode 100644 study/gitTestFork/file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ echo "Изменение 3 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git commit -m "Изменение 3 в branch1"
[branch1 a41959d] Изменение 3 в branch1
 1 file changed, 1 insertion(+), 1 deletion(-)

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$
```

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ echo "Изменение 1 в branch2" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git commit -m "Изменение 1 в branch2"
[branch2 3c3dfa6] Изменение 1 в branch2
 1 file changed, 1 insertion(+), 1 deletion(-)

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ echo "Изменение 2 в branch2" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git commit -m "Изменение 2 в branch2"
[branch2 857bd23] Изменение 2 в branch2
 1 file changed, 1 insertion(+), 1 deletion(-)

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ echo "Изменение 3 в branch2" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git commit -m "Изменение 3 в branch2"
[branch2 442c12a] Изменение 3 в branch2
 1 file changed, 1 insertion(+), 1 deletion(-)
```

5. Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git checkout branch2
D       handiwork.ini
D       landlord.txt
Already on 'branch2'

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git merge branch1
Already up to date.
```

6. Выгрузите все изменения во всех ветках в удаленный репозиторий

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git remote add origin https://github.com/KoooV/traitor.git

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git push origin branch1
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (13/13), 1010 bytes | 144.00 KiB/s, done.
Total 13 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'branch1' on GitHub by visiting:
remote:      https://github.com/KoooV/traitor/pull/new/branch1
remote:
To https://github.com/KoooV/traitor.git
 * [new branch]      branch1 -> branch1

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git push origin branch2
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (15/15), 1.17 KiB | 150.00 KiB/s, done.
Total 15 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
remote:
remote: Create a pull request for 'branch2' on GitHub by visiting:
remote:      https://github.com/KoooV/traitor/pull/new/branch2
remote:
To https://github.com/KoooV/traitor.git
 * [new branch]      branch2 -> branch2
```

7. Проведите еще 3 коммита в ветку branch1

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch2)
$ git checkout branch1
D       handiwork.ini
D       landlord.txt
Switched to branch 'branch1'

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ echo "Изменение 4 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git commit -m "Изменение 4 в branch1"
[branch1 311d193] Изменение 4 в branch1
 1 file changed, 1 insertion(+), 1 deletion(-)

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ echo "Изменение 5 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git commit -m "Изменение 5 в branch1"
[branch1 7a78107] Изменение 5 в branch1
 1 file changed, 1 insertion(+), 1 deletion(-)

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ echo "Изменение 6 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ git commit -m "Изменение 6 в branch1"
[branch1 4a4f979] Изменение 6 в branch1
 1 file changed, 1 insertion(+), 1 deletion(-)
```

8. Склонируйте репозиторий еще раз в другую директорию

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestFork (branch1)
$ cd C:/Users/Kov/study/gitTestClone

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git clone https://github.com/KoooV/traitor.git traitor-new
Cloning into 'traitor-new'...
remote: Enumerating objects: 1382, done.
remote: Counting objects: 100% (239/239), done.
remote: Compressing objects: 100% (81/81), done.
remote: Total 1382 (delta 153), reused 165 (delta 143), pack-reused 1143 (from 1)
Receiving objects: 100% (1382/1382), 2.94 MiB | 2.25 MiB/s, done.
Resolving deltas: 100% (710/710), done.
```

9. В новом клоне репозитории сделайте 3 коммита в ветку branch1

```
KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ echo "Изменение 7 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git commit -m "Изменение 7 в branch1"
[branch1 a11af3c] Изменение 7 в branch1
 1 file changed, 1 insertion(+)
 create mode 100644 study/gitTestClone/file.txt

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ echo "Изменение 8 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git commit -m "Изменение 8 в branch1"
[branch1 5cba6d1] Изменение 8 в branch1
 1 file changed, 1 insertion(+), 1 deletion(-)

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ echo "Изменение 9 в branch1" > file.txt

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git add file.txt

KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git commit -m "Изменение 9 в branch1"
[branch1 571f327] Изменение 9 в branch1
 1 file changed, 1 insertion(+), 1 deletion(-)
```

10. Выгрузите все изменения из нового репозитория в удаленный репозиторий

```
KoV@DESKTOP-TMH3BFG MINGW64 /c/Users/Kov/study/gitTestClone (branch1)
$ git push origin branch1
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
Delta compression using up to 12 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (30/30), 2.40 KiB | 153.00 KiB/s, done.
Total 30 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 1 local object.
To https://github.com/KoooV/traitor.git
   a41959d..571f327  branch1 -> branch1
```

11. Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией –force

```
KoV@DESKTOP-TMH3BFG MINGW64 /c/USers/KoV/study/gitTestFork/traitor (branch1)
$ git pull origin branch1
From https://github.com/KoooV/traitor
 * branch            branch1    -> FETCH_HEAD
Already up to date.
```

```
KoV@DESKTOP-TMH3BFG MINGW64 /c/USers/KoV/study/gitTestFork/traitor (branch1)
$ git push -f origin branch1
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 1.04 KiB | 353.00 KiB/s, done.
Total 8 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/KoooV/traitor
   571f327..d50d121  branch1 -> branch1
```

12. Получите все изменения в новом репозитории

```
KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestClone/traitor-new (main)
$ git fetch origin
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 37 (delta 9), reused 36 (delta 9), pack-reused 1 (from 1)
Unpacking objects: 100% (37/37), 3.35 KiB | 15.00 KiB/s, done.
From https://github.com/KoooV/traitor
   a41959d..d50d121  branch1    -> origin/branch1

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestClone/traitor-new (main)
$ git checkout branch1
branch 'branch1' set up to track 'origin/branch1'.
Switched to a new branch 'branch1'

KoV@DESKTOP-TMH3BFG MINGW64 ~/study/gitTestClone/traitor-new (branch1)
$ git pull origin branch1
From https://github.com/KoooV/traitor
 * branch            branch1    -> FETCH_HEAD
Already up to date.
```

№ варианта	Репозиторий
1	https://github.com/liamg/traitor
2	https://github.com/google/model_search
3	https://github.com/gto76/python-cheatsheet
4	https://github.com/ripienaar/free-for-dev
5	https://github.com/pystardust/ytfzf
6	https://github.com/airbnb/javascript
7	https://github.com/kettanaito/naming-cheatsheet
8	https://github.com/Sairyss/domain-driven-hexagon
9	https://github.com/react-native-camera/react-native-camera
10	https://github.com/spring-projects/spring-petclinic

Контрольные вопросы

1. Какие существуют типы систем контроля версий? Приведите примеры к каждому типу.

Локальные (RCS, SCCS) — хранят изменения локально.
Централизованные (CVCS: SVN, CVS) — единый сервер-репозиторий.
Распределенные (DVCS: Git, Mercurial) — полные копии репозитория у каждого пользователя.

2. К какому типу систем контроля версий относится Git?

Git — распределенная система контроля версий (DVCS).

3. Что такое репозиторий Git?

Это хранилище данных проекта, включающее историю изменений, ветки, теги и метаданные. Располагается в папке .git.

4. Что такое коммит?

Снимок изменений в репозитории на определенный момент времени. Имеет уникальный хеш (например, a1b2c3d).

5. Что такое ветка в репозитории Git?

Указатель на определенный коммит. Позволяет изолировать разработку новой функциональности.

6. Что такое тег в репозитории Git?

Метка, закрепляющая состояние репозитория (например, версия релиза:).
Отмечает значимые коммиты.

7. Что такое слияние двух веток?

Объединение изменений из двух веток в одну. Например, слияние в.

8. Что такое конфликт в Git? Как его решить и почему они бывают?

Возникает, когда изменения в разных ветках противоречат друг другу.
Решение:
Отредактировать конфликтующие участки в файлах.
Добавить исправленные файлы (git add).
Завершить слияние (git commit).
Причина: Git не может автоматически определить, какие изменения оставить.

9. Как отменить слияние веток, если произошел конфликт?

git merge --abort

10. Для чего нужен .gitignore?

Файл, указывающий Git, какие файлы/папки игнорировать (например, временные файлы, бинарники).

11. Что делает команда git status?

Показывает состояние рабочей директории и индекса: измененные, добавленные, неотслеживаемые файлы.

12. Что делает команда git add?

Добавляет изменения из рабочей директории в индекс (staging area) для последующего коммита.

13. Что делает команда git log?

Отображает историю коммитов текущей ветки. Параметры:
--oneline — краткий вывод.
--graph — визуализация веток.

14. Что делает команда git diff?

Показывает различия:
Между рабочим каталогом и индексом: git diff.
Между индексом и последним коммитом: git diff --staged.

15. Что делает команда git show?

git show a1b2c3d

16. Что делает команда git stash?

git stash pop # Восстановить изменения

17. Как добавить новую директорию в Git?

Git не отслеживает пустые папки. Чтобы добавить директорию, создайте в ней файл (например, .gitkeep).

18. Что сделает команда "git branch" без какого-либо параметра?

Выводит список локальных веток. Текущая ветка помечена звездочкой (*).

19. Чем отличаются команды "git push" и "git pull"?

git push — отправляет локальные изменения в удаленный репозиторий.
git pull — забирает изменения из удаленного репозитория и объединяет с локальным.

20. Что означает статус файла untracked в выводе команды git status?

Файл существует в рабочей директории, но не добавлен в Git (не отслеживается).

21. Что означает статус файла new в выводе команды git status?

Файл добавлен в индекс (git add), но еще не закоммичен.

22. Что означает статус файла modified в выводе команды git status?

Файл изменен после добавления в индекс, но изменения не добавлены в новый коммит.

23. Чем отличается master и origin/master?

master — локальная ветка.
origin/master — удаленная ветка на сервере (origin — имя удаленного репозитория).

24. Как узнать, кто автор строчки в файле, используя систему Git?

git blame file.txt -L 5,10 # Покажет автора строк 5-10

25. Как отменить действие команды "git add" на файл?

git reset file.txt

26. Как привести измененный файл в начальное состояние (до изменения)?

git restore file.txt

27. Как сделать ветку с названием my_branch?

git branch my_branch

28. Как сделать коммит для ветки my_branch?

git checkout my_branch
git add .

git commit -m "Message"

29. Как удалить локальную ветку my_branch?

git checkout my_branch
git add .
git commit -m "Message"

30. Как исправить ошибку "fatal: The current branch
my_branch has no upstream branch", возникающую при вводе git push?

git push --set-upstream origin my_branch

31. Как удалить все untracked файлы?

git clean -fd # -f: force, -d: директории

32. Что такое GitHub?

Платформа для хостинга Git-репозиториев с инструментами для совместной работы (Issues, Pull Requests).

33. Что такое форк репозитория?

Копия чужого репозитория на вашем аккаунте GitHub. Позволяет вносить изменения без прямого доступа к исходному репозиторию.

34. Что такое pull request?

Запрос на слияние ваших изменений из форка или ветки в основной репозиторий. Позволяет обсудить код перед мержем.