

The background of the entire page is a green circuit board pattern. It features a complex network of white lines representing circuit traces, with various electronic components like chips and pads depicted in white. The pattern is dense and covers the entire area, with a white rectangular box in the upper left containing text.

DAM 1

Exercici 52_01.

Control de versions

Soufian El Hajouji

ÍNDIX

Pregunta 1. La meva història de por.....	2
Pregunta 2. Comparem sistemes de control de versions.....	2
Pregunta 3. Quantes versions guardem?.....	5
Pregunta 4. Configuració global.....	6
Pregunta 5. Ajut d'algunes comandes interessants.....	6
Pregunta 7. Resum de comandes.....	11
Pregunta 8. Comptem objectes.....	12
Pregunta 9. Una mica de pràctica.....	13
Pregunta 11. L'art de la línia de comandes.....	21

Pregunta 1. La meva història de por

Descriu un cas concret de la teva experiència que t'hagis trobat amb un o més dels problemes descrits a la secció Justificació. Fes-ho en els termes indicats, per exemple, si ha estat entre molts treballadors o de manera telemàtica.

Concreta molt. Per exemple, descriu la convenció de noms que has fet servir per cada versió.

Una historia de miedo que he tenido y sigo teniendo, es que yo trabajo con múltiples ordenadores (**Muchas ubicaciones**), con uno de casa y uno de clase. Lo que ocurre es que al trabajar en máquinas virtuales, debo ir actualizando las máquinas para que vaya sincronizado y ha habido veces que he actualizado a la inversa perdiendo el contenido nuevo. Espero que VirtualBox en el futuro pueda poner alguna versión de la que se vaya sintonizando con tu cuenta y no ir usando la memoria USB para actualizar las máquinas con su contenido, ya que también pierdo tiempo.

Pregunta 2. Comparem sistemes de control de versions

Agafa els diferents exemples de sistemes de control de versions que hi apareixen a la secció Definició.

Crea una fitxa per cadascun d'ells amb el màxim que trobis dels següents punts:

- *nom del cvs*
- *url del projecte*
- *llicència*
- *descripció: pots copiar&enganxar la descripció breu que hi apareix a cada pàgina, si vols, directament en l'idioma original.*
- *bondats: com que hi ha prou competència entre cvs, les pàgines solen oferir de manera resumida les característiques diferenciadores del seu producte, fins i tot comparant-los amb altres.*
- *Inclou-les a la fitxa. Pot ser de manera directa (en l'idioma original) o més treballada, tractant de descriure amb les teves paraules què volen dir.*
- *impressió: una descripció de la teva primera impressió respecte el cvs*
- *Potser voldràs llegir algun article o blog per oferir una idea més informada. Et recomano, per exemple, aquest article de smashingmagazine*

1. SVN (subversion):

- **URL del proyecto:** <https://subversion.apache.org/>
- **Licencia:** Apache
- **Descripción:** Subversion (también conocido como SVN) es un sistema de control de versiones centralizado, diseñado para manejar archivos y directorios a través del tiempo.
- **Bondades:** SVN es fácil de usar y permite un seguimiento preciso de los cambios realizados en un proyecto. Además, tiene una amplia documentación y es compatible con una gran variedad de herramientas.

- **Impresión:** Creo que SVN es un sistema de control de versiones muy sólido y confiable. Aunque es centralizado en lugar de distribuido, es fácil de usar y permite un seguimiento preciso de los cambios realizados en un proyecto. Además, tiene una amplia documentación y es compatible con una gran variedad de herramientas.

2. Perforce

- **URL del proyecto:** <https://www.perforce.com/>
- **Licencia:** Propietaria
- **Descripción:** Perforce es un sistema de control de versiones que ofrece herramientas para la gestión de cambios, seguimiento de problemas y colaboración en proyectos de software. Ofrece un alto rendimiento en la gestión de grandes conjuntos de archivos y soporte para una amplia gama de plataformas y sistemas operativos.
- **Bondades:** Entre las características diferenciadoras de Perforce se incluyen la alta velocidad y eficiencia en la gestión de grandes conjuntos de archivos, la integración con diferentes herramientas de desarrollo y la capacidad de personalizar y adaptar el sistema a las necesidades de cada equipo.
- **Impresión:** Mi primera impresión sobre Perforce es que es un sistema de control de versiones muy completo y robusto, especialmente en la gestión de grandes conjuntos de archivos. Me gusta que ofrezca integración con diferentes herramientas de desarrollo y que sea personalizable para adaptarse a las necesidades específicas de cada equipo. Sin embargo, el hecho de que sea una herramienta propietaria puede ser un factor a considerar para algunas empresas o desarrolladores.

3. CVS (Concurrent Versions System)

- **URL del proyecto:** <https://www.nongnu.org/cvs/>
- **Licencia:** GPL (GNU General Public License)
- **Descripción:** CVS es un sistema de control de versiones de código abierto utilizado para la gestión y seguimiento de cambios en archivos de código fuente y otros documentos. CVS permite a los desarrolladores trabajar en equipo y realizar un seguimiento de las versiones de un proyecto de software en diferentes etapas de desarrollo. Aunque CVS es un sistema de control de versiones antiguo y ha sido reemplazado en gran medida por sistemas más avanzados como Git y SVN, todavía se usa en algunos proyectos.
- **Bondades:** Las características diferenciadoras de CVS incluyen su simplicidad y facilidad de uso, su capacidad para manejar grandes conjuntos de archivos y su capacidad para funcionar en una amplia variedad de sistemas operativos y plataformas. También cuenta con una gran cantidad de documentación y soporte en línea, debido a su larga trayectoria en el mundo del software.
- **Impresión:** Mi primera impresión sobre CVS es que es un sistema de control de versiones sólido y fácil de usar, aunque algo anticuado. Me gusta su capacidad para manejar grandes conjuntos de archivos y la gran cantidad de documentación y soporte que existe en línea. Sin embargo, dado que CVS es un sistema más antiguo, puede no ser la mejor opción para proyectos más

grandes o complejos que requieren funciones más avanzadas de control de versiones.

4. **Git:**

- **URL del proyecto:** <https://git-scm.com/>
- **Licencia:** GPLv2
- **Descripción:** Git es un sistema de control de versiones distribuido de código abierto, diseñado para manejar todo, desde proyectos pequeños hasta muy grandes con velocidad y eficiencia.
- **Bondades:** Git es muy popular y ampliamente utilizado. Permite trabajar con ramas de forma fácil y rápida, lo que facilita el trabajo en equipo y la implementación de nuevas funcionalidades. Además, es compatible con múltiples plataformas y sistemas operativos.
- **Impresión:** Me parece un sistema de control de versiones muy potente y ampliamente usado. Tiene muchas características útiles, como el manejo de ramas y la capacidad de trabajar de forma distribuida. Además, es compatible con diferentes plataformas y sistemas operativos.

5. **Mercurial:**

- **URL del proyecto:** <https://www.mercurial-scm.org/>
- **Licencia:** GPL
- **Descripción:** Mercurial es un sistema de control de versiones distribuido, que se enfoca en la velocidad y la facilidad de uso.
- **Bondades:** Mercurial es muy fácil de aprender y usar. Es muy flexible y permite trabajar con ramas de forma rápida y sencilla. Además, es compatible con diferentes plataformas y sistemas operativos.
- **Impresión:** Me parece que Mercurial es un sistema de control de versiones muy sencillo y fácil de aprender. Aunque no es tan ampliamente utilizado como Git o SVN, tiene características muy útiles, como la capacidad de trabajar con ramas de forma rápida y sencilla. Además, es compatible con diferentes plataformas y sistemas operativos.

6. **Bazaar**

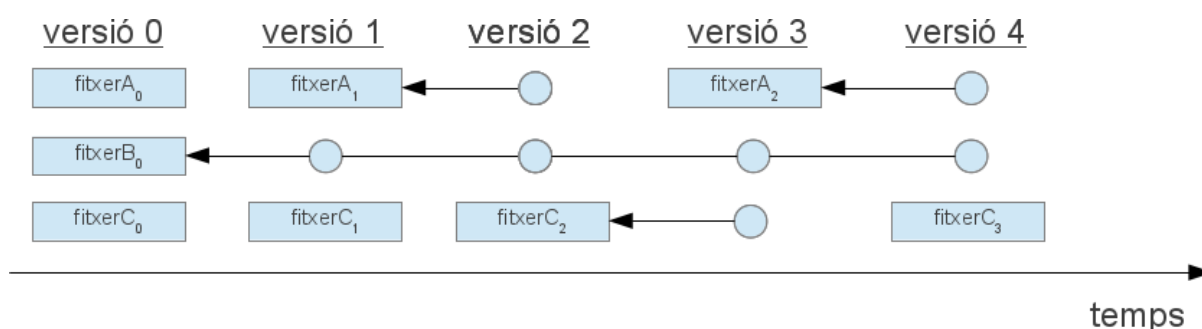
- **URL del proyecto:** <https://bazaar.canonical.com/>
- **Licencia:** GPL
- **Descripción:** Bazaar es un sistema de control de versiones distribuido que permite el seguimiento de los cambios realizados en un proyecto. Es fácil de usar y tiene una gran cantidad de características útiles, como la capacidad de trabajar con ramas de forma rápida y sencilla.
- **Bondades:** Entre las características diferenciadoras de Bazaar se incluyen la facilidad de uso, la capacidad de trabajar con proyectos de gran tamaño y la compatibilidad con diferentes plataformas y sistemas operativos.
- **Impresión:** Mi primera impresión sobre Bazaar es que es un sistema de control de versiones muy completo y fácil de usar. Me gusta que sea compatible con diferentes plataformas y que permita trabajar con proyectos de gran tamaño.

7. Darcs

- **URL del projecte:** <https://darcs.net/>
- **Licència:** GPL
- **Descripció:** Darcs es un sistema de control de versiones distribuido que permite el seguimiento de los cambios realizados en un proyecto de forma eficiente y rápida. Es fácil de usar y tiene una gran cantidad de características útiles, como la capacidad de trabajar con ramas de forma rápida y sencilla.
- **Bondades:** Entre las características diferenciadoras de Darcs se incluyen la facilidad de uso, la eficiencia en la gestión de cambios y la capacidad de trabajar con proyectos de gran tamaño.
- **Impresión:** Mi primera impresión sobre Darcs es que es un sistema de control de versiones muy interesante y eficiente. Me gusta que sea fácil de usar y que permita trabajar con proyectos de gran tamaño. Además, me parece muy útil que tenga la capacidad de trabajar con ramas de forma rápida y sencilla.

Pregunta 3. Quantes versions guardem?

Recorda l'esquema que varem veure a Introducció.



Completa la taula següent, indicant quants fitxers (no enllaços) són guardats realment per Git a cada versió d'aquest exemple:

versió	nombre de fitxers guardats
0	3
1	2
2	1
3	1
4	1

Pregunta 4. Configuració global

Un cop disposis de git instal·lat i configurat al teu sistema (Instal·lació i configuració), crida la següent instrucció a la consola:

git config --list

La resposta a aquest exercici és el resultat d'aquesta consola. Revisa que les dades siguin les que esperes.

Un cop disposis de git instal·lat i configurat al teu sistema (Instal·lació i configuració), crida la següent instrucció a la consola:

git config --list

La resposta a aquest exercici és el resultat d'aquesta consola. Revisa que les dades siguin les que esperes.

```
general@introprgvm:~$ git config --list
user.name=Soufian El Hajouji
user.email=cf22soufian.el@iesjoandaustria.org
```

Pregunta 5. Ajut d'algunes comandes interessants

Obté ajuda per les següents opcions de git:

- *clone*
- *init*
- *add*
- *mv*
- *reset*
- *rm*
- *log*
- *status*
- *checkout*
- *commit*

Per cadascuna d'aquestes opcions, afegix al document:

- *la comanda per obtenir l'ajut*
- *l'ajut generat per git. Només cal que indiqueu la breu descripció que apareix a la secció Name i la descripció que apareix a la secció Description.*

clone:

Comanda per obtenir ajuda: `git clone --help`

Ajuda generada per git:

```
GIT-CLONE(1)                                Git Manual                                GIT-CLONE(1)

NAME
    git-clone - Clone a repository into a new directory

SYNOPSIS
    git clone [--template=<template directory>]
    [-l] [-s] [--no-hardlinks] [-q] [-n] [--bare] [--mirror]
    [-o <names>] [-b <name>] [-u <upload-pack>] [--reference <repository>]
    [--associate] [--separate-git-dir <git dir>]
    [--depth <depth>] [--[no-]single-branch] [--no-tags]
    [--recurse-submodules[=<pathspec>]] [--[no-]shallow-submodules]
    [--[no-]remote-submodules] [--jobs <n>] [--sparse]
    [--filter=<filter>] [--] <repository>
    [<directory>]

DESCRIPTION
    Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository (visible using git branch --remotes), and creates and checks out an initial branch that is forked from the cloned repository's currently active branch.

    After the clone, a plain git fetch without arguments will update all the remote-tracking branches, and a git pull without arguments will in addition merge the remote master branch into the current master branch, if any (this is untrue when "--single-branch" is given; see below).

    This default configuration is achieved by creating references to the remote branch heads under refs/remotes/origin and by initializing remote.origin.url and remote.origin.fetch configuration variables.
```

init:

Comanda per obtenir ajuda: git init --help

Ajuda generada per git:

```
GIT-INIT(1)                                Git Manual                                GIT-INIT(1)

NAME
    git-init - Create an empty Git repository or reinitialize an existing one

SYNOPSIS
    git init [-q | --quiet] [--bare] [--template=<template directory>]
    [--separate-git-dir <git dir>] [--object-format=<format>]
    [-b <branch-name>] [--initial-branch=<branch-name>]
    [--shared[=<permissions>]] <directory>

DESCRIPTION
    This command creates an empty Git repository - basically a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files. An initial branch without any commits will be created (see the --initial-branch option below for its name).

    If the $GIT_DIR environment variable is set then it specifies a path to use instead of ./.git for the base of the repository.

    If the object storage directory is specified via the $GIT_OBJECT_DIRECTORY environment variable then the shl directories are created underneath - otherwise the default $GIT_DIR/objects directory is used.

    Running git init in an existing repository is safe. It will not overwrite things that are already there. The primary reason for rerunning git init is to pick up newly added templates (or to move the repository to another place if --separate-git-dir is given).
```

add:

Comanda per obtenir ajuda: git add --help

Ajuda generada per git:

```
general@introprgvmc ~
GIT-ADD(1)                                Git Manual                                GIT-ADD(1)

NAME
    git-add - Add file contents to the index

SYNOPSIS
    git add [--verbose] [-v] [--dry-run] [-n] [--force] [-f] [--interactive] [-i] [--patch] [-p]
    [--edit] [-e] [--no-poll] [--[no-]ignore-removal] [--update] [-u]
    [--intent-to-add] [-N] [--refresh] [--ignore-errors] [--ignore-missing] [--renormalize]
    [--chmod=(+|-).x] [--paths-from-file=<file>] [--paths-from-file-nul]
    [--] [<paths>...]

DESCRIPTION
    This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

    The "index" holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the contents of the next commit. Thus after making any changes to the working tree, and before running the commit command, you must use the add command to add any new or modified files to the index.

    This command can be performed multiple times before a commit. It only adds the content of the specified file(s) at the time the add command is run; if you want subsequent changes included in the next commit, then you must run git add again to add the new content to the index.

    The git status command can be used to obtain a summary of which files have changes that are staged for the next commit.

    The git add command will not add ignored files by default. If any ignored files were explicitly specified on the command line, git add will fail with a list of ignored files. Ignored files reached by directory recursion or filename globbing performed by Git (quote your globs before the shell) will be silently ignored. The git add command can be used to add ignored files with the -f (force) option.

    Please see git-commit(1) for alternative ways to add content to a commit.
```

mv:

Comanda per obtenir ajuda: git mv --help Ajuda generada per git:

```
GIT-MV(1)                                Git Manual                                GIT-MV(1)

NAME
    git-mv - Move or rename a file, a directory, or a symlink

SYNOPSIS
    git mv <options>... <args>...

DESCRIPTION
    Move or rename a file, directory or symlink.

    git mv [-v] [-f] [-n] [-k] <source> <destination>
    git mv [-v] [-f] [-n] [-k] <source> ... <destination directory>

    In the first form, it renames <source>, which must exist and be either a file, symlink or directory, to <destination>. In the second form, the last argument has to be an existing directory; the given sources will be moved into this directory.

    The index is updated after successful completion, but the change must still be committed.
```


reset:

Comanda per obtenir ajuda: `git reset --help`

Ajuda generada per git:

```
NAME
    git-reset - Reset current HEAD to the specified state

SYNOPSIS
    git reset [-q] [<tree-ish>] [--] [<pathspec>...]
    git reset [-q] [--pathspec-from-file=<file>] [--pathspec-file-null] [<tree-ish>]
    git reset [--patch] [-p] [<tree-ish>] [--] [<pathspec>...]
    git reset [--soft] [--mixed [-N]] [--hard] [--merge] [--keep] [-q] [<commit>]

DESCRIPTION
    In the first three forms, copy entries from <tree-ish> to the index. In the last form, set the current branch head (HEAD) to <commit>, optionally modifying index and working tree to match. The <tree-ish>/<commit> defaults to HEAD in all forms.

    git reset [-q] [<tree-ish>] [--] [<pathspec>...], git reset [-q] [--pathspec-from-file=<file>] [--pathspec-file-null] [<tree-ish>]
    These forms reset the index entries for all paths that match the <pathspec> to their state at <tree-ish>. (It does not affect the working tree or the current branch.)

    This means that git reset <pathspec> is the opposite of git add <pathspec>. This command is equivalent to git restore [--source=<tree-ish>] --staged <pathspec>....

    After running git reset <pathspec> to update the index entry, you can use git-restore(1) to check the contents out of the index to the working tree. Alternatively, using git-restore(1) and specifying a commit with --source, you can copy the contents of a path out of a commit to the index and to the working tree in one go.

    git reset [--patch] [-p] [<tree-ish>] [--] [<pathspec>...]
    Interactively select hunks in the difference between the index and <tree-ish> (defaults to HEAD). The chosen hunks are applied in reverse to the index.

    This means that git reset -p is the opposite of git add -p, i.e. you can use it to selectively reset hunks. See the "Interactive Mode" section of git-add(1) to learn how to operate the --patch mode.

    git reset [<mode>] [<commit>]
    This form resets the current branch head to <commit> and possibly updates the index (resetting it to the tree of <commit>) and the working tree depending on <mode>. If <mode> is omitted, defaults to --mixed. The <mode> must be one of the following:

    --soft
        Does not touch the index file or the working tree at all (but resets the head to <commit>, just like all modes do). This leaves all your changed files "Changes to be committed", as git status would put it.

    --mixed
        Resets the index but not the working tree (i.e., the changed files are preserved but not marked for commit) and reports what has not been updated. This is the default action.

        If -N is specified, removed paths are marked as intent-to-add (see git-add(1)).

    --hard
        Resets the index and working tree. Any changes to tracked files in the working tree since <commit> are discarded.

    --merge
        Resets the index and updates the files in the working tree that are different between <commit> and HEAD, but keeps those which are different between the index and working tree (i.e. which have changes which have not been added). If a file that is different between <commit> and the index has unstaged changes, reset is aborted.

    --keep
        Resets index entries and updates files in the working tree that are different between <commit> and HEAD. If a file that is different between <commit> and HEAD has local changes, reset is aborted.

    --[no]-recurse-submodules
        When the working tree is updated, using --recurse-submodules will also recursively reset the working tree of all active submodules according to the commit recorded in the superproject, also setting the submodules' HEAD to be detached at that commit.

    See "Reset, restore and revert" in git(1) for the differences between the three commands.
```

rm:

Comanda per obtenir ajuda: `git rm --help`

Ajuda generada per git:

```
GIT-RM(1)                                Git Manual                                GIT-RM(1)

NAME
    git-rm - Remove files from the working tree and from the index

SYNOPSIS
    git rm [-f] [--force] [-n] [--] [--cached] [--ignore-unmatch]
           [--quiet] [--pathspec-from-file=<file>] [--pathspec-file-null]
           [--] [<pathspec>...]

DESCRIPTION
    Remove files matching pathspec from the index, or from the working tree and the index. git rm will not remove a file from just your working directory. (There is no option to remove a file only from the working tree and yet keep it in the index; use /bin/rm if you want to do that.) The files being removed have to be identical to the tip of the branch, and no updates to their contents can be staged in the index, though that default behavior can be overridden with the -f option. When --cached is given, the staged content has to match either the tip of the branch or the file on disk, allowing the file to be removed from just the index.
```

log:

Comanda per obtenir ajuda: `git log --help`

Ajuda generada per git:

```
GIT-LOG(1)                                Git Manual                                GIT-LOG(1)

NAME
    git-log - Show commit logs

SYNOPSIS
    git log [<options>] [<revision range>] [--] [<path>...]

DESCRIPTION
    Shows the commit logs.

    List commits that are reachable by following the parent links from the given commit(s), but exclude commits that are reachable from the one(s) given with a ^ in front of them. The output is given in reverse chronological order by default.

    You can think of this as a set operation. Commits reachable from any of the commits given on the command line form a set, and then commits reachable from any of the ones given with ^ in front are subtracted from that set. The remaining commits are what comes out in the command's output. Various other options and paths parameters can be used to further limit the result.

    Thus, the following command:

        $ git log foo bar "baz"

    means "list all the commits which are reachable from foo or bar, but not from baz".

    A special notation "<commit1>..<commit2>" can be used as a short-hand for "<commit1> <commit2>". For example, either of the following may be used interchangeably:

        $ git log origin..HEAD
        $ git log HEAD "origin"

    Another special notation is "<commit1>...<commit2>" which is useful for merges. The resulting set of commits is the symmetric difference between the two operands. The following two commands are equivalent:

        $ git log A B --not $(git merge-base --all A B)
        $ git log A...B

    The command takes options applicable to the git-rev-list(1) command to control what is shown and how, and options applicable to the git-diff(1) command to control how the changes each commit introduces are shown.
```

status:

Comanda per obtenir ajuda: `git status --help`

Ajuda generada per git:

```
GIT-STATUS(1)                             Git Manual                             GIT-STATUS(1)

NAME
    git-status - Show the working tree status

SYNOPSIS
    git status [<options>...] [--] [<pathspec>...]

DESCRIPTION
    Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked by git (and are not ignored by gitignore(5)). The first are what you would commit by running git commit; the second and third are what you could commit by running git add before running git commit.
```

checkout:

Comanda per obtenir ajuda: git checkout --help

Ajuda generada per git:

```
GIT-CHECKOUT(1)                                Git Manual                                GIT-CHECKOUT(1)

NAME
    git-checkout - Switch branches or restore working tree files

SYNOPSIS
    git checkout [-q] [-f] [-m] [<branch>]
    git checkout [-q] [-f] [-m] --detach [<branch>]
    git checkout [-q] [-f] [-m] [--detach] <commit>
    git checkout [-q] [-f] [-m] [[-b|-B|--orphan] <new_branch>] [<start_point>]
    git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] [--] [<pathspec>...]
    git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] --pathspec-from-file=<file> [--pathspec-file=<file>]
    git checkout [-p|--patch] [<tree-ish>] [--] [<pathspec>...]

DESCRIPTION
    Updates files in the working tree to match the version in the index or the specified tree. If no pathspec was given, git checkout will also update HEAD to set the specified branch as the current branch.

    git checkout <branch>
    To prepare for working on <branch>, switch to it by updating the index and the files in the working tree, and by pointing HEAD at the branch. Local modifications to the files in the working tree are kept, so that they can be committed to the <branch>.

    If <branch> is not found but there does exist a tracking branch in exactly one remote (call it <remote>) with a matching name and --no-guess is not specified, treat as equivalent to

        $ git checkout -b <branch> --track <remote>/<branch>

    You could omit <branch>, in which case the command degenerates to "check out the current branch", which is a glorified no-op with rather expensive side-effects to show only the tracking information, if exists, for the current branch.

    git checkout -b|-B <new_branch> [<start_point>]
    Specifying -b causes a new branch to be created as if git-branch(1) were called and then checked out. In this case you can use the --track or --no-track options, which will be passed to git branch. As a convenience, --track without -b implies branch creation; see the description of --track below.

    If -B is given, <new_branch> is created if it doesn't exist; otherwise, it is reset. This is the transactional equivalent of

        $ git branch -f <branch> [<start_point>]
        $ git checkout <branch>

    that is to say, the branch is not reset/created unless "git checkout" is successful.

    git checkout --detach [<branch>], git checkout [--detach] <commit>
    Prepare to work on top of <commit>, by detaching HEAD at it (see "DETACHED HEAD" section), and updating the index and the files in the working tree. Local modifications to the files in the working tree are kept, so that the resulting working tree will be the state recorded in the commit plus the local modifications.

    When the <commit> argument is a branch name, the --detach option can be used to detach HEAD at the tip of the branch (git checkout <branch> would check out that branch without detaching HEAD).

    When the <commit> argument is a branch name, the --detach option can be used to detach HEAD at the tip of the branch (git checkout <branch> would check out that branch without detaching HEAD).

    Omitting <branch> detaches HEAD at the tip of the current branch.

    git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] [--] [<pathspec>...], git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] --pathspec-from-file=<file> [--pathspec-file=<file>]
    Overwrite the contents of the files that match the pathspec. When the <tree-ish> (most often a commit) is not given, overwrite working tree with the contents in the index. When the <tree-ish> is given, overwrite both the index and the working tree with the contents at the <tree-ish>.

    The index may contain unmerged entries because of a previous failed merge. By default, if you try to check out such an entry from the index, the checkout operation will fail and nothing will be checked out. Using -f will ignore these unmerged entries. The contents from a specific side of the merge can be checked out of the index by using --ours or --theirs. With -m, changes made to the working tree file can be discarded to re-create the original conflicted merge result.

    git checkout (-p|--patch) [<tree-ish>] [--] [<pathspec>...]
    This is similar to the previous mode, but lets you use the interactive interface to show the "diff" output and choose which hunks to use in the result. See below for the description of --patch option.
```

commit:

Comanda per obtenir ajuda: git commit --help

Ajuda generada per git:

```
NAME
    git-commit - Record changes to the repository

SYNOPSIS
    git commit [-a] [--interactive] [--patch] [-s] [-w] [-u=<mode>] [--amend]
    [--dry-run] [--<c | -C | --fixup | --squash>] <commit>
    [-F <file>] [-m <msg>] [--reset-author] [--allow-empty]
    [--allow-empty-message] [--no-verify] [-e] [--author=<author>]
    [--date=<date>] [--cleanup=<mode>] [--no-status]
    [-i] [-o] [--pathspec-from-file=<file>] [--pathspec-file=<file>]
    [-S<keyid>] [--] [<pathspec>...]

DESCRIPTION
    Create a new commit containing the current contents of the index and the given log message describing the changes. The new commit is a direct child of HEAD, usually the tip of the current branch, and the branch is updated to point to it (unless no branch is associated with the working tree, in which case HEAD is "detached" as described in git-checkout(1)).

    The content to be committed can be specified in several ways:

    1. by using git-add(1) to incrementally "add" changes to the index before using the commit command (Note: even modified files must be "added");

    2. by using git-rm(1) to remove files from the working tree and the index, again before using the commit command;

    3. by listing files as arguments to the commit command (without --interactive or --patch switch), in which case the commit will ignore changes staged in the index, and instead record the current content of the listed files (which must already be known to Git);

    4. by using the -a switch with the commit command to automatically "add" changes from all known files (i.e. all files that are already listed in the index) and to automatically "rm" files in the index that have been removed from the working tree, and then perform the actual commit;

    5. by using the --interactive or --patch switches with the commit command to decide one by one which files or hunks should be part of the commit in addition to contents in the index, before finalizing the operation. See the "Interactive Mode" section of git-add(1) to learn how to operate these modes.

    The --dry-run option can be used to obtain a summary of what is included by any of the above for the next commit by giving the same set of parameters (options and paths).

    If you make a commit and then find a mistake immediately after that, you can recover from it with git reset.
```

Pregunta 6. Configuració inicial

Un cop hakis fet la creació d'un repositori segons (Creació d'un repositori: init), crida la següent instrucció a la consola:

\$ git config --list

Respon a les següents preguntes:

quina sortida t'ha generat la comanda anterior

què voldrà dir el valor de core.bare?

1. Quina sortida t'ha generat la comanda anterior

```
general@introprgvm:~/projecte$ git config --list
user.name=Soufian El Hajouji
user.email=cf22soufian.el@iesjoandaustria.org
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
general@introprgvm:~/projecte$
```

2. Què voldrà dir el valor de core.bare?

El valor de core.bare se refereix a si un repositori se configura com un repositori bare o no. Un repositori bare és un repositori que no té un arbre de treball associat, lo que significa que no es poden fer canvis en el mateix. Si el valor de core.bare és "true", llavors el repositori està configurat com un repositori bare. Si és "false" o no està definit, llavors el repositori no és bare.

Pregunta 7. Resum de comandes

Creat un petit resum de les comandes que han aparegut en aquesta introducció a Git. Inclou les comandes comentades a partir de suggeriments del propi Git, com ara reset.

Pensa-ho com una xuleta o cheat sheet que serveixi de referència ràpida pel Git.

Fes servir un format tabular amb les capçaleres:

Comanda (ex. commit -m «comentari»)

Petita descripció (ex. "registra els canvis a stage associant-los un comentari")

Exemple (ex. "\$ git commit -m "Registre inicial")

Comanda	Petita descripció	Exemple
git init	Crea un nou repositori git	git init
git clone	Clona un repositori git existent	git clone https://github.com/user/repo.git

git add	Afegir canvis del directori de treball	git add file.txt
git commit	Registra els canvis al repositori local	git commit -m "Missatge del commit"
git status	Mostra l'estat del directori de treball	git status
git log	Mostra l'historial de commits	git log
git diff	Mostra les diferències entre els canvis del staging i WD	git diff
git checkout	Canvia de branca o restaura fitxers	git checkout -b new-branch
git reset	Desfà els canvis del staging o d'un commit específic	git reset HEAD file.txt
git rm	Elimina fitxers del directori de treball i del repositori	git rm file.txt
git mv	Canvia el nom o la ubicació d'un fitxer	git mv file.txt new_location/
git remote	Gestiona connexions a repositoris remots	git remote add origin https://github.com/user/repo.git
git push	Puja els canvis al repositori remot	git push origin main
git pull	Baixa els canvis del repositori remot	git pull origin main

Pregunta 8. Comptem objectes

A partir dels continguts de Passar a stage, crea un nou repositori, per exemple a partir de /tmp.

Pas 0. Compta el nombre d'objectes que està guardant Git en aquest moment:

\$ git count-objects

Guarda el resultat de la comanda anterior.

Pas 1. Crea-hi un fitxer anomenat test.txt però de moment no l'afegeixis a Git.

Compta un altre cop quants objectes guarda Git.

Pas 2. Afegeix test.txt a Git però de moment no facis commit.

Torna a comptar quants objectes guarda Git.

Pas 3. Fes el primer commit i torna a comptar el nombre d'objectes que està guardant.

Pas 4. Modifica el fitxer test.txt de la següent manera:

\$ git config --list > test.txt

Abans de fer commit, torna a revisar quants objectes guarda Git

Pas 5. Fes commit i torna a comptar quants objectes en portes.

Emplena la següent taula:

pas	objectes	kilobytes
0	0	0
1	1	12
2	2	24
3	3	36
4	3	36
5	4	48

Pregunta 9. Una mica de pràctica

Practicarem ara els continguts que hem vist a Branques.

Per la realització d'aquest exercici hauràs de prendre nota de totes les comandes que llencis i continguts de fitxers que toquis.

Segueix les instruccions i prepara els resultats finals que seran en forma de bundle més un document amb les descripcions de les passes i les comandes/continguts de fitxer.

Les instruccions et deixaran lliures alguns elements (ex. noms, alguns continguts de fitxers, missatges de commit) És molt recomanable que les realitzis sobre un GNU/Linux o, com a mínim, un sistema compatible POSIX per facilitar-te la captura.

Instruccions

- 1. Personalitza adequadament el teu Git.*
- 2. Crea un repositori Git*
- 3. Afegeix al directori de treball un fitxer amb un contingut generat per la comanda \$ ip a (o l'equivalent al teu sistema) des de la teva màquina.*
- 4. Fes el primer commit*
- 5. Copia el contingut del teu fitxer de configuració .gitconfig a un fitxer dins del directori de treball i afegeix-lo al control de versions.*
- 6. Crea un nou fitxer on afegeixis la data i hora del sistema (date), afegeix-lo a stage i comprova l'estat del projecte.*

7. *Modifica el primer fitxer tot afegint-hi al final la data i hora del sistema i comprova l'estat del projecte.*
8. *Registra tots els canvis i torna a comprovar l'estat del projecte.*
9. *Consulta l'historial de canvis fins el moment de tot el projecte.*
10. *Configura el teu projecte de manera que Git no intenti gestionar els .class. Realitza les accions que consideris per comprovar que no ho està fent.*
11. *Realitza les accions que trobis necessàries per demostrar com es pot veure els canvis realitzats al contingut d'un fitxer que encara no s'han passat a stage.*
12. *Com al punt anterior, però aquest cop demostra com comprovar els canvis realitzats en un fitxer respecte el darrer commit.*
13. *Consulta ara l'històric de canvis d'un dels fitxers incloent les diferències de cada versió respecte de l'anterior.*
14. *Realitza les comandes necessàries per demostrar com es pot tornar a l'estat del darrer commit d'un determinat fitxer.*
15. *Crea una nova branca, modifica-hi un dels fitxers i registra els canvis. Comprova que els canvis hi són a la nova branca però no a la branca mestra. Finalment torna a la branca mestra, fusiona els canvis de la nova branca i elimina la nova branca. Comprova que l'has eliminada.*
16. *Crea una nova branca i realitza els canvis necessaris sobre algun dels fitxers, de manera que les dues línies de desenvolupament (la nova branca i la mestra) divergeixin. Aconsegueix que en intentar fusionar els canvis a la branca mestra algun dels fitxers modificats sigui fusionat automàticament però algun altre requereixi modificacions manuals. Finalment hauria de quedar tots els canvis fusionats a la branca mestra i l'altra branca quedar eliminada.*
17. *Guarda tot el contingut de la branca mestra en un paquet git.*

1. Personalitzem el Git executant les següents comandes:

```
git config --global user.name "Soufian"
```

```
git config --global user.email proni10102002@gmail.com
```

2. Creem un repositori Git amb la comanda:

```
mkdir proyecto
```

```
cd proyecto
```

```
git init
```

3. Generem un fitxer amb la comanda:

```
ip a > fitxer.txt
```

Aquesta comanda ens genera un fitxer "fitxer.txt" amb les dades de la interfície de xarxa de la màquina.

4. Afegim el fitxer al repositori i fem el primer commit:

```
git add fitxer.txt
```

```
git commit -m "Primer commit"
```

5. Copiem el contingut del fitxer de configuració ".gitconfig" al directori de treball:

```
cp ~/.gitconfig gitconfig.txt
```

Ara afegim el fitxer al repositori i fem el commit:

```
git add gitconfig.txt
```

```
git commit -m "Afegit fitxer gitconfig.txt"
```

6. Creem un fitxer amb la data i hora del sistema amb la comanda:

```
date > data.txt
```

Afegim el fitxer a stage:

```
git add data.txt
```

I comprovem l'estat del projecte:

```
git status
```

7. Modifiquem el fitxer original "fitxer.txt" amb la data i hora del sistema:

```
date >> fitxer.txt
```

I comprovem l'estat del projecte:

```
git status
```

8. Registrem els canvis amb la comanda:

```
git add .
```

```
git commit -m "Afegida data i hora al fitxer.txt"
```

I comprovem l'estat del projecte:

```
git status
```

9. Configurem el projecte perquè Git no intenti gestionar els fitxers ".class" amb la comanda:

```
echo "*.class" >> .gitignore
```


Per comprovar que no estem gestionant aquests fitxers, podem crear-ne un amb la comanda:

```
touch fitxer.class
```

I comprovar l'estat del projecte amb la comanda:

```
git status
```

- 10. Per veure els canvis realitzats al contingut d'un fitxer que encara no s'ha afegit a stage, podem utilitzar la comanda:**

```
git diff fitxer.txt
```

- 11. Per comprovar els canvis realitzats en un fitxer respecte el darrer commit, podem utilitzar la comanda:**

```
git diff HEAD fitxer.txt
```

- 12. Consultem l'historial de canvis fins el moment de tot el projecte amb la comanda:**

```
git log -p fitxer.txt
```

per sortir

```
q
```

- 13. Per tornar a l'estat del darrer commit d'un fitxer determinat, s'utilitza la comanda:**

```
git checkout fitxer.txt
```

- 14. Per crear una nova branca, modificar un fitxer i fer un commit en la nova branca, comprovar els canvis a la nova branca però no a la branca mestra, tornar a la branca mestra, fusionar els canvis de la nova branca i eliminar la nova branca, es realitzen les següents comandes:**

```
git branch nova_branca
```

```
git checkout nova_branca
```

```
git checkout master
```

```
git merge nova_branca
```

```
git branch -d nova_branca
```

- 15. Per crear una nova branca i fer que les dues línies de desenvolupament divergeixin, es realitzen les següents comandes:**

git checkout -b nova_branca

git add .

git commit -m "canvis en la nova branca"

git checkout master

git merge nova_branca

git add .

git commit -m "fusio de canvis en la nova branca"

git branch -d nova_branca

16. guardar tot el contingut de la branca mestra en un paquet git, s'utilitza la comanda:

git bundle create nom_paquet.bundle master

```

general@introprgvm:~$ git config --global user.name "Soufian"
git config --global user.email proni10102002@gmail.com
general@introprgvm:~$ mkdir proyecto
cd proyecto
git init
pista: Using 'master' as the name for the initial branch. This default branch name
pista: is subject to change. To configure the initial branch name to use in all
pista: of your new repositories, which will suppress this warning, call:
pista:
pista: git config --global init.defaultBranch <name>
pista:
pista: Names commonly chosen instead of 'master' are 'main', 'trunk' and
pista: 'development'. The just-created branch can be renamed via this command:
pista:
pista: git branch -m <name>
S'ha inicialitzat un dipòsit buit del Git en /home/general/proyecto/.git/
general@introprgvm:~/proyecto$ ip a > fitxer.txt
general@introprgvm:~/proyecto$ git add fitxer.txt
git commit -m "Primer commit"
[master (comissió arrel) c87fc41] Primer commit
1 file changed, 12 insertions(+)
create mode 100644 fitxer.txt
general@introprgvm:~/proyecto$ cp ~/.gitconfig gitconfig.txt
general@introprgvm:~/proyecto$ git add gitconfig.txt
git commit -m "Afegeix fitxer gitconfig.txt"
[master 389fef0] Afegeix fitxer gitconfig.txt
1 file changed, 3 insertions(+)
create mode 100644 gitconfig.txt
general@introprgvm:~/proyecto$ date > data.txt
general@introprgvm:~/proyecto$ git add data.txt
general@introprgvm:~/proyecto$ git status
En la branca master
Canvis a cometre:
    (use "git restore --staged <file>..." to unstage)
        fitxer nou:      data.txt

general@introprgvm:~/proyecto$ date >> fitxer.txt
general@introprgvm:~/proyecto$ git status
En la branca master
Canvis a cometre:
    (use "git restore --staged <file>..." to unstage)
        fitxer nou:      data.txt

Canvis no «staged» per a cometre:
    (useu «git add <fitxer>...» per a actualitzar què es cometrà)
    (use "git restore <file>..." to discard changes in working directory)
        modificat:      fitxer.txt

general@introprgvm:~/proyecto$ git add .
git commit -m "Afegeix data i hora al fitxer.txt"
[master f5697d8] Afegeix data i hora al fitxer.txt
2 files changed, 2 insertions(+)
create mode 100644 data.txt
general@introprgvm:~/proyecto$ git status
En la branca master
No hi ha res a cometre, l'arbre de treball està net
general@introprgvm:~/proyecto$ echo "*.class" >> .gitignore
general@introprgvm:~/proyecto$ touch fitxer.class
general@introprgvm:~/proyecto$ git status
En la branca master
Fitxers no seguits:
    (useu «git add <fitxer>...» per a incloure-ho en la comissió)
        .gitignore

no hi ha res afegit a cometre però hi ha fitxers no seguits (useu «git add» per a seguir-los)
general@introprgvm:~/proyecto$ git diff fitxer.txt
general@introprgvm:~/proyecto$ git diff HEAD fitxer.txt
general@introprgvm:~/proyecto$ git log -p fitxer.txt
commit f5697d85dbd20dd109a1a2ab57c86df1d6a3f007 (HEAD -> master)

```

```

general@introprgvm:~/proyecto$ git log -p fitxer.txt
commit f5697d85dbd20dd109a1a2ab57c86dfd6a3f007 (HEAD -> master)
Author: Soufian <pronil10102002@gmail.com>
Date: Sat Apr 29 17:12:09 2023 +0200

    Afegida data i hora al fitxer.txt

diff --git a/fitxer.txt b/fitxer.txt
index 2f6bb60..e4200bc 100644
--- a/fitxer.txt
+++ b/fitxer.txt
@@ -10,3 +10,4 @@
     valid_lft 83819sec preferred_lft 83819sec
     inet6 fe80::a00:27ff:fe4d:37c4/64 scope link noprefixroute
     valid_lft forever preferred_lft forever
+disabte, 29 d'abril de 2023, 17:11:58 CEST

commit c87fc411fd12355fee83fa2b842bb2cb854d5fdf
Author: Soufian <pronil10102002@gmail.com>
Date: Sat Apr 29 17:11:31 2023 +0200

    Primer commit

diff --git a/fitxer.txt b/fitxer.txt
new file mode 100644
index 0000000..2f6bb60
--- /dev/null
+++ b/fitxer.txt
@@ -0,0 +1,12 @@
general@introprgvm:~/proyecto$ git checkout fitxer.txt
S'ha actualitzat 0 camins des de l'índex
general@introprgvm:~/proyecto$ git branch nova_branca
git checkout nova_branca
git checkout master
git merge nova_branca
git branch -d nova_branca
S'ha canviat a la branca «nova_branca»
S'ha canviat a la branca «master»
Ja està al dia.
S'ha suprimit la branca nova_branca (era f5697d8).
general@introprgvm:~/proyecto$ git checkout -b nova_branca
git add .
git commit -m "canvis en la nova branca"
git checkout master
git merge nova_branca
git add .
git commit -m "fusio de canvis en la nova branca"
git branch -d nova_branca
S'ha canviat a la branca nova «nova_branca»
[nova branca c2dc613] canvis en la nova branca
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
S'ha canviat a la branca «master»
S'estan actualitzant f5697d8..c2dc613
Fast-forward
 .gitignore | 1 +
 1 file changed, 1 insertion(+)
create mode 100644 .gitignore
En la branca master
no hi ha res a cometre, l'arbre de treball està net
S'ha suprimit la branca nova_branca (era c2dc613).
general@introprgvm:~/proyecto$ git bundle create nom_paquet.bundle master
S'estan enumerant els objectes: 13, fet.
S'estan comptant els objectes: 100% (13/13), fet.
S'estan comprimint els objectes: 100% (10/10), fet.
Total 13 (delta 2), reused 0 (delta 0), pack-reused 0

```

Pregunta 10. Visualització

També existeixen eines gràfiques per visualitzar els repositoris git. Aquestes eines permeten navegar de manera més visual per l'històric de canvis.

Una de les més senzilles d'instal·lar és, potser, gitk) . Aquesta la podràs instal·lar i executar amb:

```
$ sudo apt-get install gitk
```

«...»

```
projecte$ gitk
```

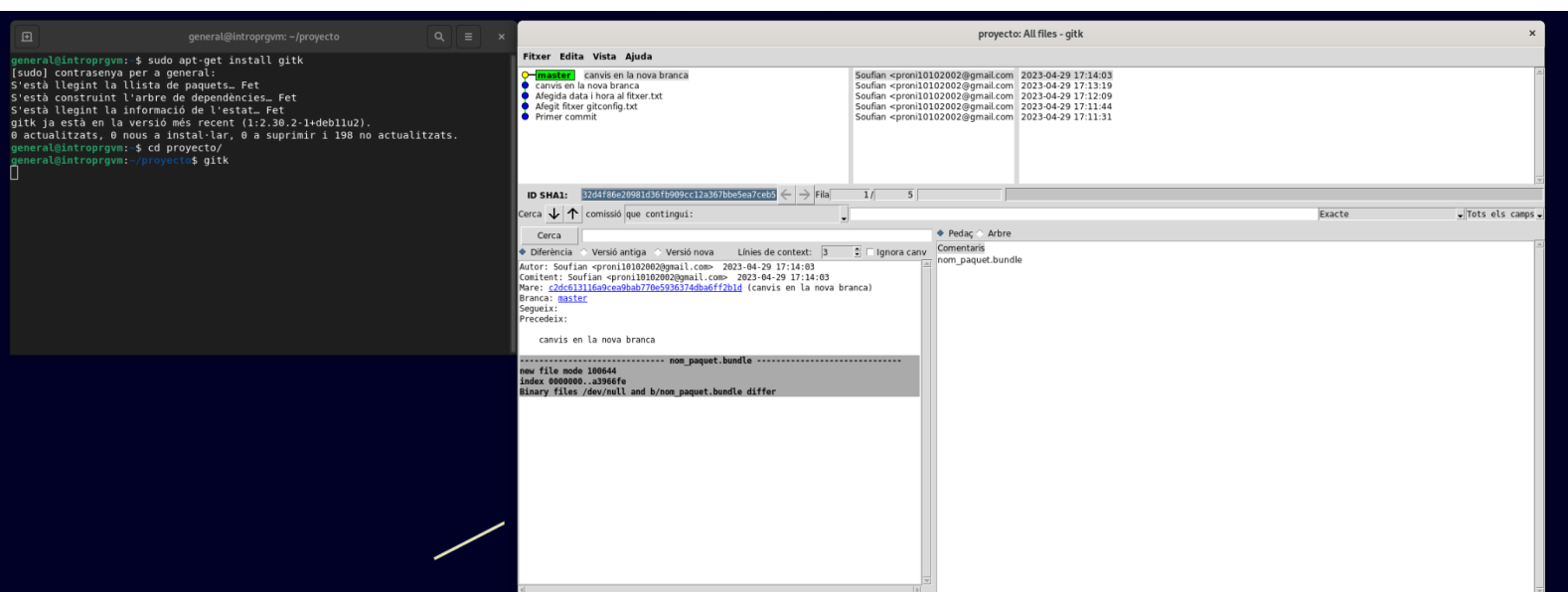
Per a aquest exercici, instal·la't l'eina gitk o alguna alternativa

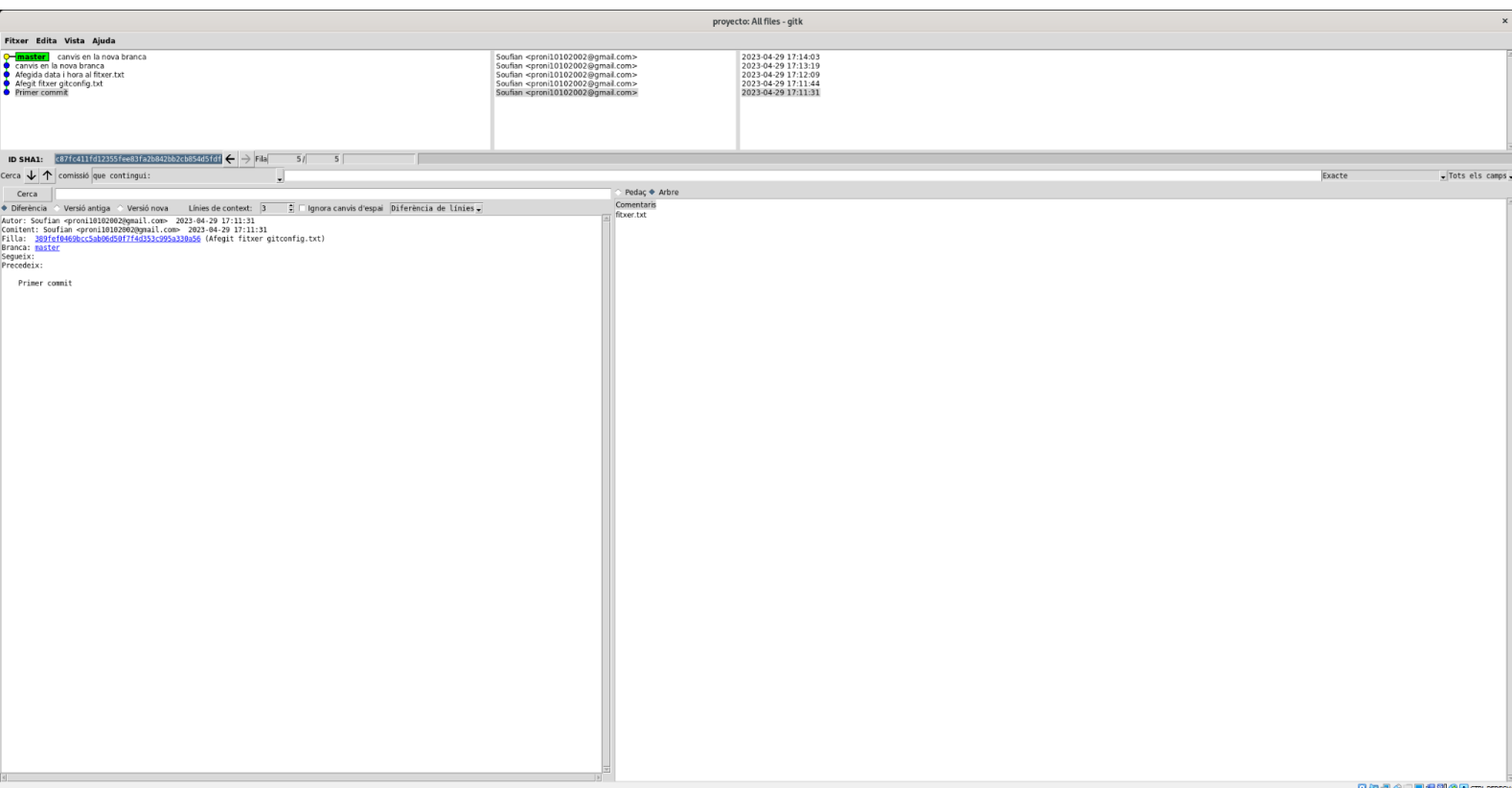
Desenvolupa un petit tutorial de la instal·lació que inclogui captures de pantalla de l'eina amb el projecte anterior.

```
sudo apt-get install gitk
```

```
cd proyecto
```

gitk





Pregunta 11. L'art de la línia de comandes

El nostre idioma no es troba entre les traduccions de The art of command line

En aquest exercici farem un pas per mirar d'arreglar aquesta situació.

Per a poder gestionar aquest exercici sense afectar en jlevy, he creat un fork al meu compte de GitHub.

#. Fes un fork d'aquest repositori amb el teu compte de github.

- 1. Clona'l al teu equip.**
- 2. Crea una nova branca i anomena-la catalan**
- 3. Afegeix-hi un nou fitxer anomenat README-ca.md amb els continguts del fitxer README.md.**
- 4. Comença a fer la traducció dels seus continguts. Com a mínim, tradueix la línia 27. Si et veus amb dificultats, pots fer servir algun traductor automàtic (no li ho direm a ningú ;)**
- 5. Un cop ho tinguis, fes els commits pertinents i puja-ho al teu repositori remot**

La resposta a aquest exercici inclourà la url del teu repositori a GitHub i captures de pantalla amb explicacions de cada pas.

<https://github.com/soufian123/the-art-of-command-line>

```
git clone git@github.com:soufian123/the-art-of-command-line.git
```

```
cd the-art-of-command-line/
```

```
git branch catalan
```

```
git checkout catalan
```

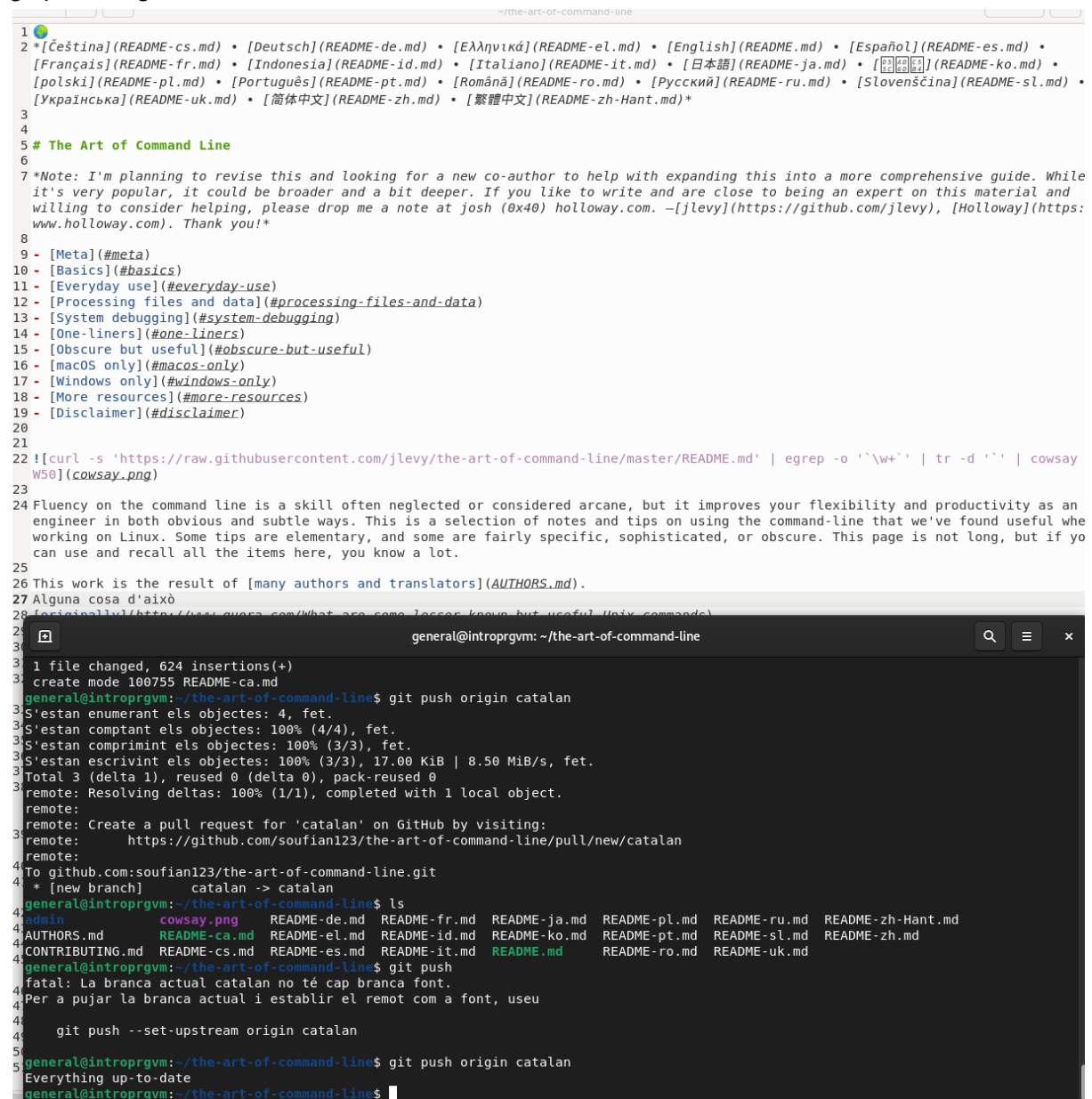
```
cp README.md README-ca.md
```

```
gedit README-ca.md &
```

```
git add README-ca.md
```

```
git commit -m "Afegida traducció de la línia 27"
```

```
git push origin catalan
```



```
1 2 * [Čeština] (README-cs.md) • [Deutsch] (README-de.md) • [Ελληνικά] (README-el.md) • [English] (README.md) • [Español] (README-es.md) •  
3 [Français] (README-fr.md) • [Indonesia] (README-id.md) • [Italiano] (README-it.md) • [日本語] (README-ja.md) • [한국어] (README-ko.md) •  
4 [polski] (README-pl.md) • [Português] (README-pt.md) • [Română] (README-ro.md) • [Русский] (README-ru.md) • [Slovenščina] (README-sl.md) •  
5 [Українська] (README-uk.md) • [简体中文] (README-zh.md) • [繁體中文] (README-zh-Hant.md) *  
6  
7 # The Art of Command Line  
8  
9 *Note: I'm planning to revise this and looking for a new co-author to help with expanding this into a more comprehensive guide. While  
10 it's very popular, it could be broader and a bit deeper. If you like to write and are close to being an expert on this material and  
11 willing to consider helping, please drop me a note at josh (0x40) holloway.com. -[jlevy](https://github.com/jlevy), [Holloway](https:  
12 www.holloway.com). Thank you! *  
13  
14 - [Meta] (#meta)  
15 - [Basics] (#basics)  
16 - [Everyday use] (#everyday-use)  
17 - [Processing files and data] (#processing-files-and-data)  
18 - [System debugging] (#system-debugging)  
19 - [One-liners] (#one-liners)  
20 - [Obscure but useful] (#obscure-but-useful)  
21 - [macOS only] (#macos-only)  
22 - [Windows only] (#windows-only)  
23 - [More resources] (#more-resources)  
24 - [Disclaimer] (#disclaimer)  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091
```