Leandro Lika
Lucca Alt
Nillan Sivarasa

Task 1

b)

```
lennsco@lennsco:~/Desktop/DPIContext$ ./chatsh.sec init Alice
Initialized empty Git repository in /home/lennsco/Desktop/DPIContext/Alice/.git/
lennsco@lennsco:~/Desktop/DPIContext$ git cat-file --batch-all-objects --batch-check
fatal: not a git repository (or any of the parent directories): .git
  128 lennsco@lennsco:~/Desktop/DPIContext$ cd Alice/
lennsco@lennsco:~/Desktop/DPIContext/Alice$ git cat-file --batch-all-objects --batch-check
4b825dc642cb6eb9a060e54bf8d69288fbee4904 tree 0
bbc69c28aa4589ecb8faf61d43b2680f6eb34b11 commit 181
lennsco@lennsco:~/Desktop/DPIContext/Alice$ git log
commit bbc69c28aa4589ecb8faf61d43b2680f6eb34b11 (HEAD -> Alice)
Author: Lennsco <leandro09.lika@gmail.com>
Date:   Wed Apr 2 23:23:14 2025 +0000

    Alice joined
lennsco@lennsco:~/Desktop/DPIContext/Alice$
```

c)

```
lennsco@lennsco:~/Desktop/DPIContext$ ./chatsh.sec init Bob
Initialized empty Git repository in /home/lennsco/Desktop/DPIContext/Bob/.git/
lennsco@lennsco:~/Desktop/DPIContext$ cd Alice/
lennsco@lennsco:~/Desktop/DPIContext/Alice$ git remote add bob ../Bob
lennsco@lennsco:~/Desktop/DPIContext/Alice$ git push bob Alice
Enumerating objects: 2, done.
Counting objects: 100% (2/2), done.
Writing objects: 100% (2/2), 167 bytes | 167.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To ../Bob
 * [new branch]      Alice -> Alice
lennsco@lennsco:~/Desktop/DPIContext/Alice$ cd  ~/Desktop/DPIContext/
lennsco@lennsco:~/Desktop/DPIContext$ cd Bob/
lennsco@lennsco:~/Desktop/DPIContext/Bob$ git cat-file --batch-all-objects --batch-check
4b825dc642cb6eb9a060e54bf8d69288fbee4904 tree 0
613d4dfae9206cd716741b5bde93951b82c86ae2 commit 179
bbc69c28aa4589ecb8faf61d43b2680f6eb34b11 commit 181
lennsco@lennsco:~/Desktop/DPIContext/Bob$ git cat-file -p bbc69c28aa4589ecb8faf61d43b2680f6eb34b11
tree 4b825dc642cb6eb9a060e54bf8d69288fbee4904
author Lennsco <leandro09.lika@gmail.com> 1743636194 +0000
committer Lennsco <leandro09.lika@gmail.com> 1743636194 +0000

Alice joined
lennsco@lennsco:~/Desktop/DPIContext/Bob$
```

d)

```
  128 lennsco@lennsco:~/Desktop/DPIContext$ cd Bob/
lennsco@lennsco:~/Desktop/DPIContext/Bob$ git log --all --graph --oneline
* 613d4df (HEAD -> Bob) Bob joined
* bbc69c2 (Alice) Alice joined
lennsco@lennsco:~/Desktop/DPIContext/Bob$
```

`git log [<options>] [<revision-range>] [[-] <path>...]` will show the commit logs. This command lists commits that are reachable by following the parent links from the given commit(s), but excludes 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.

Thus, the following command, for example:

```
git log foo bar ^baz
```

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

Leandro Lika
Lucca Alt
Nillan Sivarasa

Therefore,

```
git log
```

will list all the commits which are reachable from the directory in which we currently are. Thus the commits from `/Bob`. When we run git push bob Alice, Alice's branch is transferred but Bob's local head still points to Bob's own branch, not Alice's. `Git log` without arguments then only shows commits reachable from the current `HEAD`.
One possible solution could be that we merge Alice's branch into Bob's. With git merge Alice and then git log.

Another solution would be to inspect all refs:

```
git log --all --graph --oneline
```

This will show all the commits made.

Task 2

d)

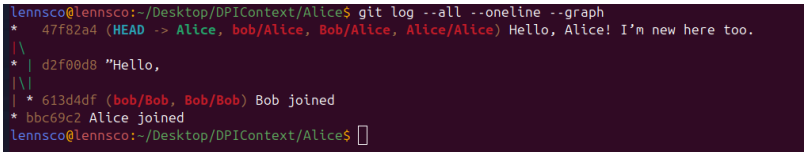

Figure 1: Alice branch

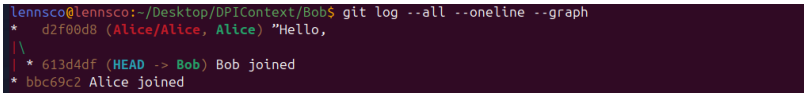

Figure 2: Bob branch

Leandro Lika
Lucca Alt
Nillan Sivarasa

Task 3

b)

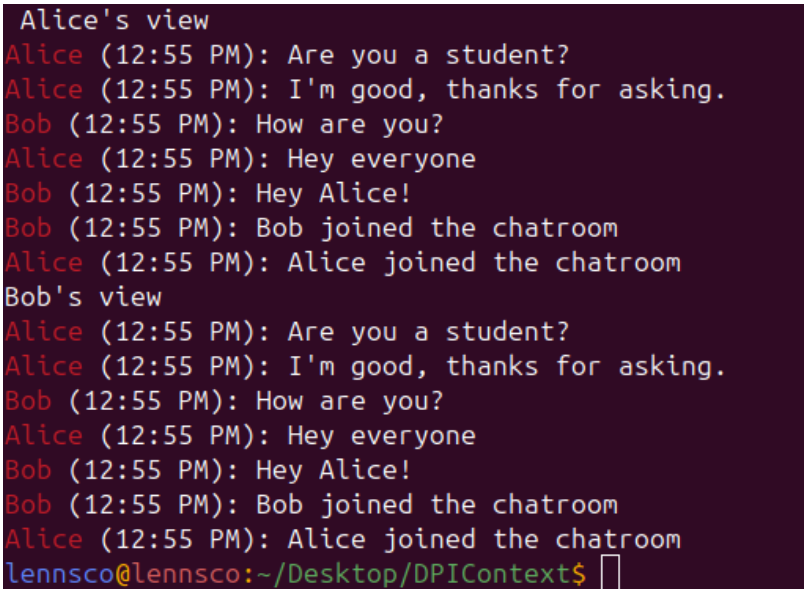

Figure 3: Output script Task3b.sec

c)

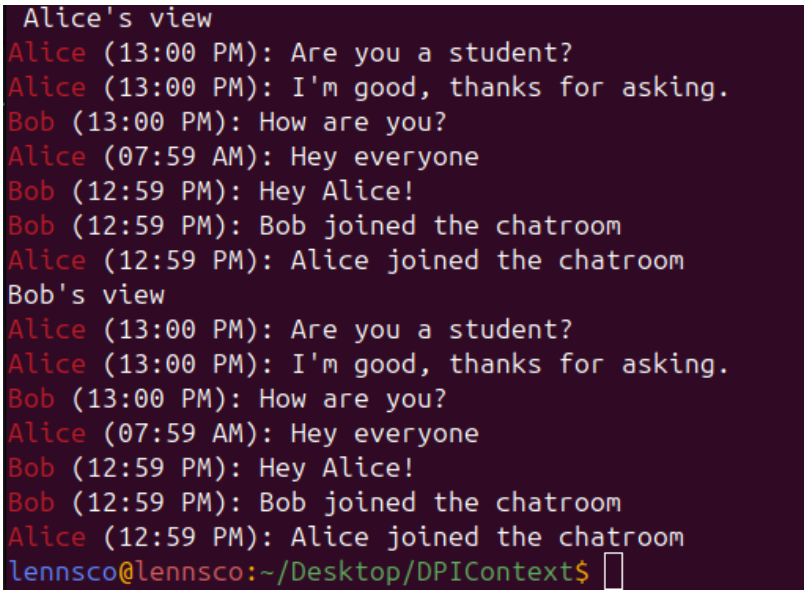

Figure 4: Output script Task3c.sec

When viewing the git log (after changing system time), does this change the order of the messages?
No it does not change the order of the messages. Git uses commit graph topology to determine the order and not timestamps. We view our chat log using our show() function, which runs ... git log −−all −−topo-order −−pretty=format:... which displays commits in an order that respects parent-child relationships. Therefore, even if Alice goes back in time, git still places her message after any commits it descends from.

d)

Scenario 1: New user Carol posts a message without first pulling from Alice

When Carol posts a message first without pulling from Alice, she will create an initial commit that is unaware of other messages such as from Alice. Her git repository is outdated. If she uses the post command it uses her local references as the commits parents. Thus git sees her message as an branch. When using git log, her message appears on a separate line of history. The messages are therefore split into separate branches and therefore not relating.

Scenario 2: Carol does pull the latest state before posting.
Here Carol does pull form all remotes before posting her message. Her repository is up to date with the latest commits from other users. When she decides to post her commit it will include those recent commits of Alice as its parents. Thus, git will know that it is the continuation of the shared conversation.

Why is it important to merge or pull the most recent commits before appending new messages in distributed scenarios?

When we observe both scenarios it becomes clear that it is beneficial to pull or merge the most recent commits before appending new messages, since it ensures that the messages / conversations are not disconnected.