**Follow steps in order. Some of the tasks need to be completed in pairs, so it works better if you sit together. Have fun ☺**

**Gloria+Riccardo**

1. In your GitHelloWorld project folder, create an empty text file named after you (i.e.: <YOUR\_NAME.TXT>).
2. Instruct Git to add it to local staging area with <git add FILE\_NAME>.
3. Commit changes locally with <git commit>. the VI editor will appear: push <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, push <ESC> to go back to command mode, and type <:wq> to write file and quit (for reference, <:> invokes command mode in VI, <w> stands for write and <q> for quit).
4. Push changes to remote repository with <git push>. This should result in the runner-up being requested to do a <git pull> before being able to commit.
5. Since you’ve been working on different files, the merge should happen without conflicts.

**Ale**

1. Refresh repository using <git pull>.
2. Verify that both Gloria's and Riccardo's commits are added to your local repository with <git log>. If not, you'll have to wait for their commits.
3. Open "Gloria.txt" and write 5 random words in it, one per line. Save it and close it.
4. Add the modified file to local staging area with <git add FILE\_NAME>.
5. Commit changes locally with <git commit>. The VI editor will appear: push <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, push <ESC> to go back to command mode, and type <:wq> to write file and quit.
6. Push changes to remote repository with <git push>.

**Michael**

1. Refresh repository using <git pull>.
2. Verify that Gloria's, Riccardo's and Ale's commits are added to your local repository with <git log>. If not, you'll have to wait for their commits.
3. Open "Riccardo.txt" and write 5 random words in it, one per line. Save and close it.
4. Add the modified file to local staging area with <git add FILE\_NAME>.
5. Commit changes locally with <git commit>. The VI editor will appear: push <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, push <ESC> to go back to command mode, and type <:wq> to write file and quit.
6. Push changes to remote repository with <git push>.

**(Stretch Task) Rachel+Ben / Gloria+Riccardo / Michael+Ale - one pair at a time, so synchronize!**

You will both race to push a modification of the same line in a file: this will create a file write conflict.

1. Refresh your local repository with <git pull> so that you have all previous changes reflected. (IMPORTANT: make sure you both execute <git pull> before moving to next steps).
2. You'll both open the file "Gloria.txt" and modify the word on the first line with any other word of your choice. Statistically this should result in a conflict (if not we forget Git and go play EuroMillion).
3. Push changes to remote repository with <git push>.
4. The runner-up should be asked to perform <git pull>. This will result in a merge conflict.
5. Examine the conflict using <git mergetool>. This will fire up a VI editor window where you can see conflicts and choose final output. Press <i> to go into insert mode and modify the file. More info on conflicts and conflict markers can be found at <https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line/>. Once you agree on the result, press <ESC> to go back to command mode, and type <:wqa> to write file and quit.
6. When done, use <git add> to stage the fix and <git commit> to commit it. The VI editor will appear: press <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, press <ESC> to go back to command mode, and type <:wq> to write file and quit.
7. Finally, use <git push> to upload changes to the remote.

**Ben**

You will create a project branch, modify a file in it, and then merge the branch into the master.

1. Make sure you refresh your repository with <git pull> so that you have all previous changes reflected locally.
2. First, create a branch (parallel development stream) named after you with the command <git branch YOUR\_NAME>
3. Switch context to it via <git checkout YOUR\_NAME>. The command <git branch -a> will show all available project branches and highlight the active one.
4. Open "Riccardo.txt" and add a random word to the list.
5. Add the modified file to local staging area with <git add FILE\_NAME>.
6. Commit changes locally with <git commit>. The VI editor will appear: press <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, press <ESC> to go back to command mode, and type <:wq> to write file and quit.
7. Move back to the master branch with <git checkout master>.
8. Merge your own branch to master using <git merge YOUR\_NAME>.
9. Push changes to remote repository with <git push> (if you get an error just follow the suggested command edit).

**Rachel**

You will merge Ben’s branch into the master.

1. Make sure you refresh your local repository with <git pull> so that you have all previous changes reflected locally (don't start before Rachel has committed her changes).
2. Run the command <git branch -a> to verify you synchronized Ben’s branch
3. Switch context to it via <git checkout YOUR\_NAME>. The command <git branch -a> will show all available project branches and highlight the active one.
4. Move back to the master branch with <git checkout master>.
5. Merge Ben’s branch to master using <git merge YOUR\_NAME>.
6. Open "Gloria.txt" and add a random word to the list.
7. Add the modified file to local staging area with <git add>.
8. Commit changes locally with <git commit>. The VI editor will appear: press <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, press <ESC> to go back to command mode, and type <:wq> to write file and quit.
9. Push changes to remote repository with <git push> (if you get an error just follow the suggested command edit).

**(Stretch Task) Any pair, but always one pair at a time**

You will both race to merge a branch that modifies the other branch's file: this will create a merge conflict.

1. Refresh your repository with <git pull> so that you have all previous changes reflected locally (IMPORTANT: make sure you both execute <git pull> before moving to next steps).
2. First, create a branch named after you with the command <git branch YOUR\_NAME>
3. Switch context to it via <git checkout YOUR\_NAME>. The command <git branch -a> will show all available project branches and highlight the active one.
4. One member of the pair will open one of the files and add a word to the list, then add it to local staging area with <git add FILE\_NAME>. The other member will remove the same file from its branch with <git rm FILE\_NAME>
5. Commit changes locally with <git commit>. The VI editor will appear: press <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, press <ESC> to go back to command mode, and type <:wq> to write file and quit.
6. Commit changes with <git commit>.
7. Move back to the master branch with <git checkout master>.
8. Merge your own branch locally to master using <git merge YOUR\_NAME>.
9. Push changes to remote repository with <git push>.
10. The runner-up should be asked to perform <git pull>. This will result in a merge conflict.
11. Examine the conflict using <git mergetool>. This will fire up a VI editor window where you can see conflicts and choose final output. Press <i> to go into insert mode and modify the file. More info on conflict markers can be found at <https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line/>. Once you agree on the result, press <ESC> to go back to command mode, and type <:wqa> to write file and quit.
12. When done, use <git add> to stage the fix and <git commit> to commit it. The VI editor will appear: press <i> to go into insert mode and uncomment all relevant lines (from email address till the end). Once done, press <ESC> to go back to command mode, and type <:wq> to write file and quit.
13. Finally, use <git push> to upload changes to the remote.

**All**

1. Refresh local repository using <git pull> so that we’re all synchronized.