Lab 1 Report

Registered under CS Lab 02

Mingzhe Wang

Hyosik Moon

Xing Li

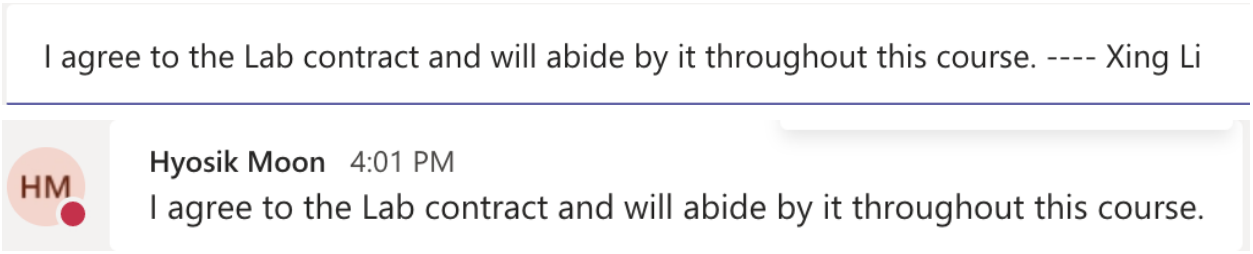
Table of Contents

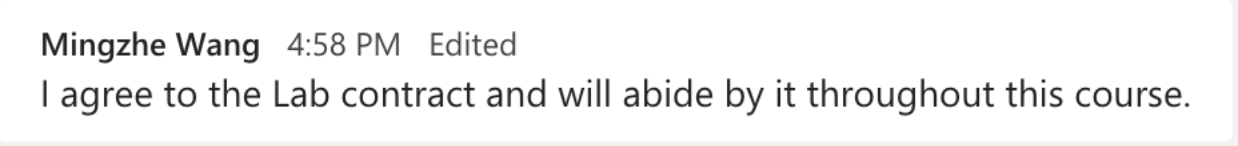
Group Information

Name	Student Number	McMaster Email	Enrolled Lab
Mingzhe Wang	400316660	wangm235@mcmaster.ca	CS Lab 02 (Registered)
Hyosik Moon	400295620	moonh8@mcmaster.ca	CS Lab 04
Xing Li	400292346	li64@mcmaster.ca	CS Lab 03

Lab Contract

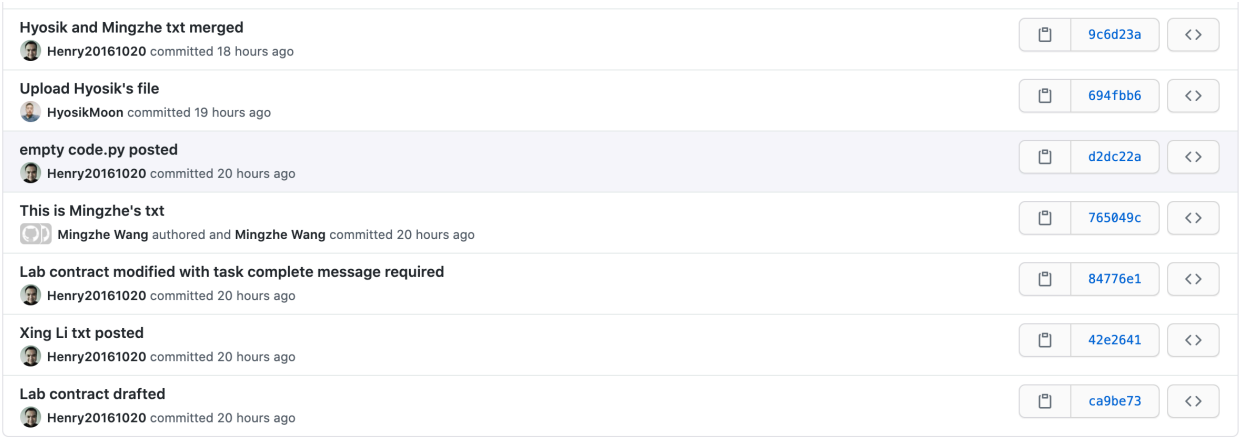
1. There is one contact member in the group who will submit the labs: Xing Li
2. Each week there is one coordinator who is in charge of the overall lab progress. The coordinator shall be the main contributor to the lab and shall dispatch the tasks with due date if needed to make sure the lab is completed efficiently. The coordinator will be rotated in the three members of the group.
3. All members shall closely monitor the team chat channel to respond to teammates as soon as possible, no later than 12 hours. When a task is finished, the owner shall send a message in the communication channel to get all members informed.
4. Two weekly meetings are scheduled. One is on Monday from 7pm to 7:30pm for general design and dividing tasks. One is on Friday from 12:30pm to 1pm for review. The Friday meeting might be cancelled if everything goes smoothly.
5. All members will use GitHub as the repository.
6. All members will use Microsoft Teams as the major communication channel.
7. When one member cannot attend the meeting or complete tasks on time, he shall inform the whole team as early as possible.





Create a Git repository

We created a repository at GitHub. All members created a file <name>.txt, write something in it and push it to the repository.



Research on Revert and Reset

Git Checkout

Functionality

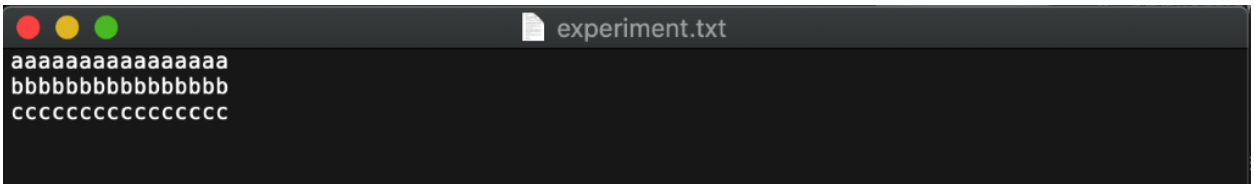
A checkout is an operation that moves the HEAD ref pointer to a specified commit.

Usage scenarios

If you have modified a file in your working tree, but haven't committed the change, then you can use git checkout to checkout a fresh-from-repository copy of the file.

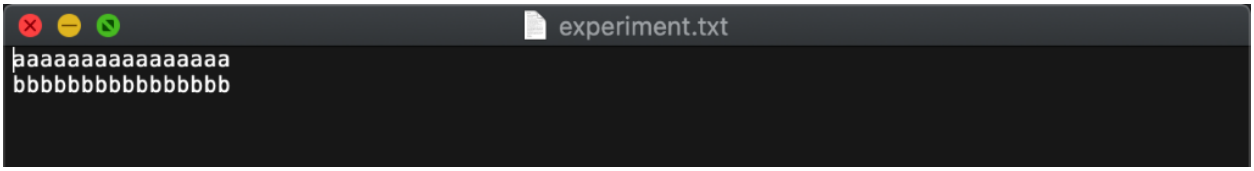
Experiment

Before:



After command:

```
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git log --oneline
46cf467 (HEAD -> master) version-c
91dde59 version-b
f2c6a40 version-a
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git checkout 91dde59
Note: switching to '91dde59'.

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 91dde59 version-b
```



Git Reset

Functionality

A reset is an operation that takes a specified commit and resets the "three trees" to match the state of the repository at that specified commit.

For git reset, there are three ways to alter the staged snapshot and/or the working directory by passing it one of the following flags:

- --soft – The staged snapshot and working directory are not altered in any way.
- --mixed – The staged snapshot is updated to match the specified commit, but the working directory is not affected. This is the default option.
- --hard – The staged snapshot and the working directory are both updated to match the specified commit.

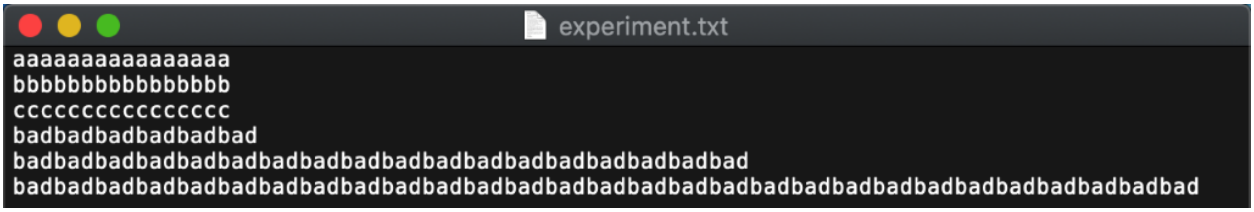
Usage scenarios

If you have made a commit, but haven't shared it with anyone else and you decide you don't want it, then you can use git reset to rewrite the history so that it looks as though you never made that commit.

Experiment

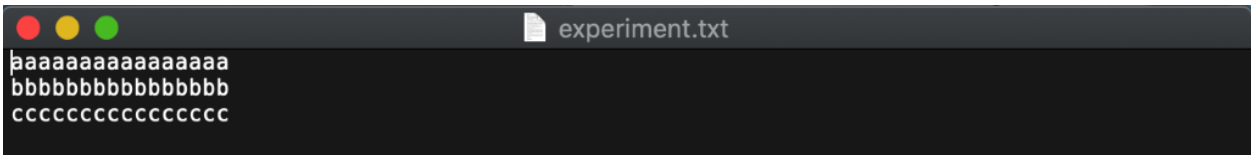
Before:

```
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git log --oneline
fa6934a (HEAD -> master) version-bad
46cf467 version-c
91dde59 version-b
f2c6a40 version-a
```



After command:

```
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git reset --hard 46cf467
HEAD is now at 46cf467 version-c
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git log --oneline
46cf467 (HEAD -> master) version-c
91dde59 version-b
f2c6a40 version-a
```



Git Revert

Functionality

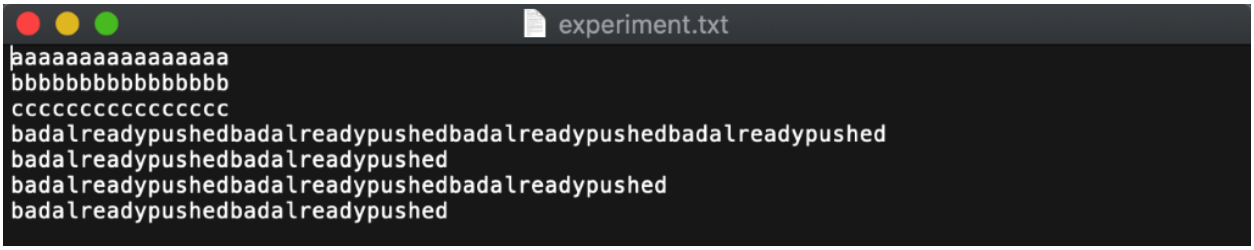
A revert is an operation that takes a specified commit and creates a new commit which inverses the specified commit.

Usage scenarios

If a commit has been made somewhere in the project's history, and you later decide that the commit is wrong and should not have been done, then git revert is the tool for the job. It will undo the changes introduced by the bad commit, recording the "undo" in the history.

Experiment

Before:



```
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git log --oneline
f8d4266 (HEAD -> master) version-badalreadypushed
46cf467 (origin/master) version-c
91dde59 version-b
f2c6a40 version-a
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/myron0220/git_experiement.git
   46cf467..f8d4266  master -> master
```

Mingzhe Wang version-badalreadypushed

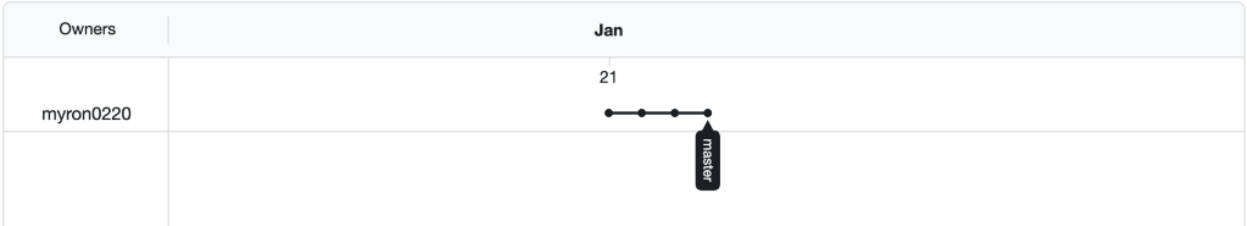
0 contributors

7 lines (7 sloc) | 230 Bytes

```
1  aaaaaaaaaaaaaaa
2  bbbbbbbbbbbbbbb
3  ccccccccccccccc
4  badalreadypushedbadalreadypushedbadalreadypushedbadalreadypushed
5  badalreadypushedbadalreadypushed
6  badalreadypushedbadalreadypushedbadalreadypushed
7  badalreadypushedbadalreadypushed
```

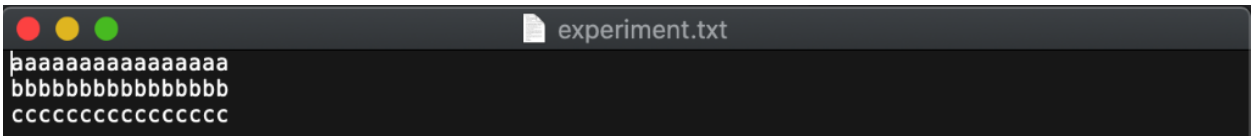
Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

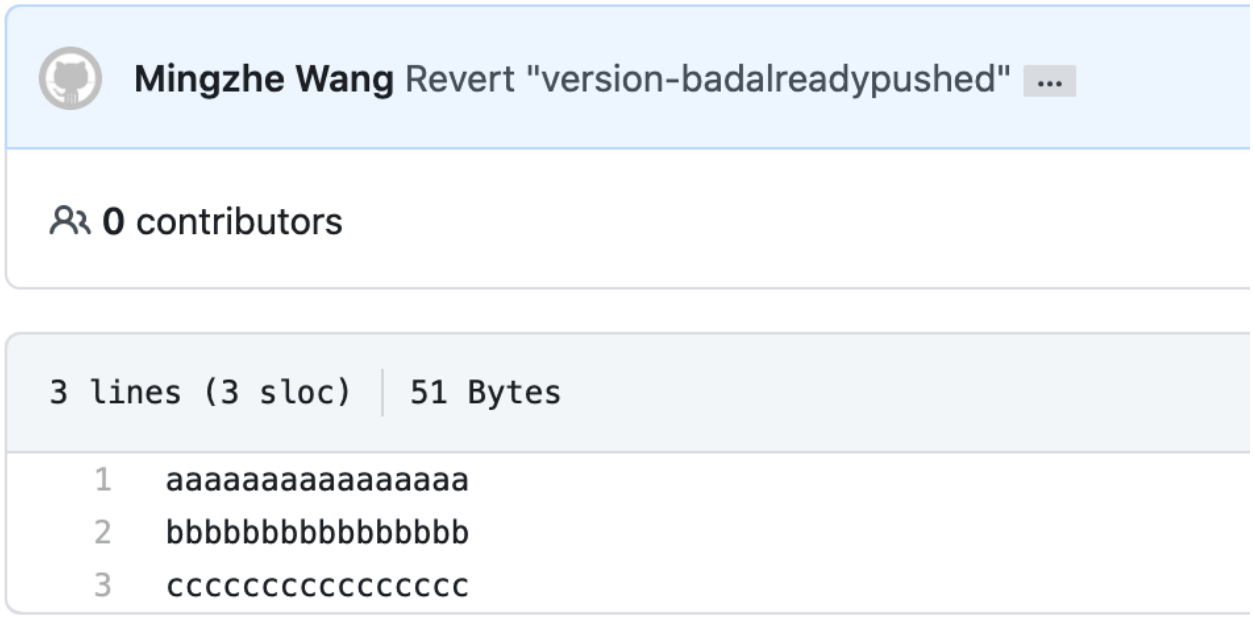


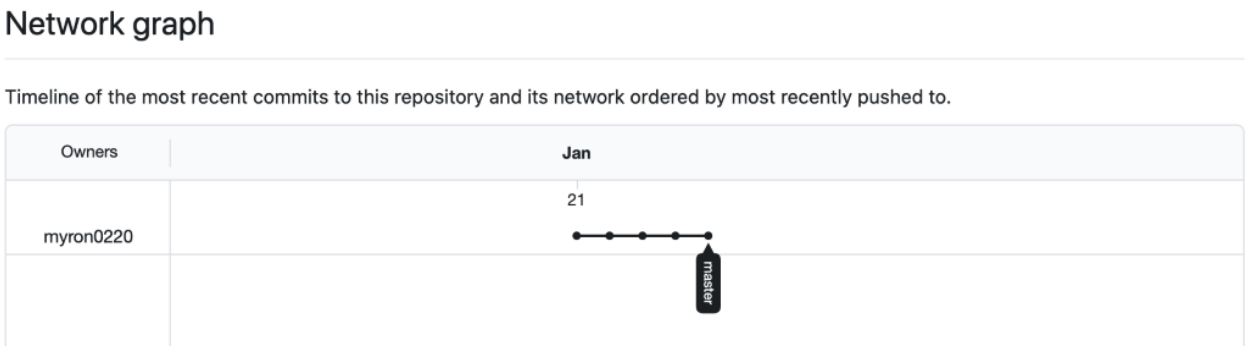
After command:

```
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git log --oneline
f8d4266 (HEAD -> master, origin/master) version-badalreadypushed
46cf467 version-c
91dde59 version-b
f2c6a40 version-a
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git revert f8d4266
[master 2a47d8a] Revert "version-badalreadypushed"
 1 file changed, 4 deletions(-)
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git log --oneline
2a47d8a (HEAD -> master) Revert "version-badalreadypushed"
f8d4266 (origin/master) version-badalreadypushed
46cf467 version-c
91dde59 version-b
f2c6a40 version-a
```




```
(base) Mingzhes-MacBook-Pro:git_experiement kidsama$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/myron0220/git_experiement.git
   f8d4266..2a47d8a  master -> master
```





Summary

Checkout and reset are generally used for making local or private 'undos'. They modify the history of a repository that can cause conflicts when pushing to remote shared repositories.

Revert is considered a safe operation for 'public undos' as it creates new history which can be shared remotely and doesn't overwrite history remote team members may be dependent on.

Update “code.py”

We used different approaches to deal with the conflict.

Approach 1

Delete the “.git” folder and load again. This is not a recommended approach, but we did experiment this at the very beginning.

> Sorry, I just uploaded my file. When I had tried to upload my file, I got an error.. It took about an hour to fix it.. I don't know why it didn't work... I just deleted the .git folder and reload it, and then it worked.

Approach 2

Here’s another issue. When a merge was required, the terminal was disconnected. Then in the next time to pull, we were reminded to finish the previous merge first.

```
MacBooks-MBP:2XB3 macbook$ git pull
error: You have not concluded your merge (MERGE_HEAD exists).
hint: Please, commit your changes before merging.
fatal: Exiting because of unfinished merge.
MacBooks-MBP:2XB3 macbook$
```

The instruction below was adopted to fix. (Retrieved from https://stackoverflow.com/questions/11646107/you-have-not-concluded-your-merge-merge-head-exists)

1. Undo the merge and pull again.

   To undo a merge:

   git merge --abort [Since git version 1.7.4]

   git reset --merge [prior git versions]
2. Resolve the conflict.
3. Don't forget to add and commit the merge.
4. git pull now should work fine.

VIM is the default editor, so the VIM commands were also searched. (i for insert mode, Esc for normal mode, Retrieved from https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started)

```
MacBooks-MBP:lab1 macbook$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/Henry20161020/2XB3
   4e8af91..a0004fe  master     -> origin/master
Updating 4e8af91..a0004fe
error: Your local changes to the following files would be overwritten by merge:
        lab1/code.py
Please commit your changes or stash them before you merge.
Aborting
MacBooks-MBP:lab1 macbook$
```

After the file “code.py” was modified, “git pull” failed because “git commit” had not been done yet.

```
MacBooks-MBP:lab1 macbook$ git add code.py
MacBooks-MBP:lab1 macbook$ git commit -m "code.py modified by Xing Li"
[master fc4072b] code.py modified by Xing Li
 1 file changed, 13 insertions(+)
MacBooks-MBP:lab1 macbook$ git pull
Auto-merging lab1/code.py
CONFLICT (content): Merge conflict in lab1/code.py
Automatic merge failed; fix conflicts and then commit the result.
MacBooks-MBP:lab1 macbook$
```

After the file “code.py” was committed, there was still conflict between origin and local. Then we tried to branch the current local work and force pull the origin.

```
MacBooks-MBP:lab1 macbook$ git branch henry
MacBooks-MBP:lab1 macbook$ git fetch origin master
From https://github.com/Henry20161020/2XB3
 * branch            master     -> FETCH_HEAD
MacBooks-MBP:lab1 macbook$ git reset --hard FETCH_HEAD
HEAD is now at a0004fe Add code.py file, Hyosik
MacBooks-MBP:lab1 macbook$ git clean -df
Removing ~$b 1 Report.docx
MacBooks-MBP:lab1 macbook$
```

Auto merge still failed, but we could see that both codes appeared in the file “code.py”.

```
MacBooks-MBP:lab1 macbook$ git merge henry
Auto-merging lab1/code.py
CONFLICT (content): Merge conflict in lab1/code.py
Automatic merge failed; fix conflicts and then commit the result.
MacBooks-MBP:lab1 macbook$ cat code.py
<<<<<<< HEAD
def are_valid_groups(students, groups):
    output = False
    student_num = len(students)
    true_num = 0
    for group in groups:
        for student in students:
            if student in group:
                true_num += 1
        if true_num == student_num:
            output = True
            break
    return output

# test
students1 = [1111, 2222, 3333]
students2 = [1111, 2222, 3334]
groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
print(are_valid_groups(students1, groups))
print(are_valid_groups(students2, groups))

=======
# Created by Xing Li
## @brief Check if every student number in the list appears in one of the groups
#  @param sl list of integers representing student numbers
#  @param gl list of strings representing group ID
#  @return True if every student number in the list appears in one of the groups, False otherwise
def are_valid_groups(sl, gl):
  for s in sl:
    if sum([1 if s in g for g in gl]) == 0:
      return False
  return True



>>>>>>> henry
MacBooks-MBP:lab1 macbook$
```

We chose to keep both code blocks, committed and pushed. It’s done.

```
MacBooks-MBP:lab1 macbook$ git add code.py
MacBooks-MBP:lab1 macbook$ gi add "Lab 1 Report.docx"
-bash: gi: command not found
MacBooks-MBP:lab1 macbook$ git add "Lab 1 Report.docx"
MacBooks-MBP:lab1 macbook$ git status
On branch master
Your branch is up to date with 'origin/master'.

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   Lab 1 Report.docx
        modified:   code.py

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

        ../.DS_Store

MacBooks-MBP:lab1 macbook$ git commit -m "Hyosik and Henry's code.py merged. Lab 1 Report updated"
[master 29bcb8d] Hyosik and Henry's code.py merged. Lab 1 Report updated
MacBooks-MBP:lab1 macbook$ git push
Enumerating objects: 17, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 395.99 KiB | 23.29 MiB/s, done.
Total 9 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/Henry20161020/2XB3.git
   a0004fe..29bcb8d  master -> master
```

In summary, we experimented different options to resolve the conflict, and finally we found that the most straightforward way is to pull the origin, edit the conflicts in an editor and commit to merge.

Approach 3

Finally, we used the most straightforward approach and successfully resolve the conflict efficiently

```
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/Henry20161020/2XB3.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git pull
error: Pulling is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git branch
* master
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 7 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
        renamed:    Lab 1/HyosikMoon.txt -> lab1/HyosikMoon.txt
        new file:   lab1/Lab 1 Report.docx
        renamed:    Lab 1/Ming zhe.txt -> lab1/Ming zhe.txt
        renamed:    Lab 1/Xing Li.txt -> lab1/Xing Li.txt
        new file:   lab1/code.py

Unmerged paths:
  (use "git add/rm <file>..." as appropriate to mark resolution)
        deleted by them: Lab 1/code.py

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:   lab1/code.py

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

(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git add lab1/code.py
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git rm Lab 1/code.py
fatal: pathspec 'Lab' did not match any files
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git rm "Lab 1/code.py"
rm 'Lab 1/code.py'
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ ls
```

```
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ ls
Lab Contract.docx       lab1
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 7 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        deleted:    Lab 1/code.py
        renamed:    Lab 1/HyosikMoon.txt -> lab1/HyosikMoon.txt
        new file:   lab1/Lab 1 Report.docx
        renamed:    Lab 1/Ming zhe.txt -> lab1/Ming zhe.txt
        renamed:    Lab 1/Xing Li.txt -> lab1/Xing Li.txt
        new file:   lab1/code.py

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

(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git commit -m "Merge all three"
[master 03c1454] Merge all three
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.10 KiB | 1.10 MiB/s, done.
Total 8 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To https://github.com/Henry20161020/2XB3.git
   25068d2..03c1454  master -> master
(base) Mingzhes-MacBook-Pro:2XB3 kidsama$
```

Player and Adversary

First Round (Player = Hyosik, Adversary = Xing, Writer = Hyosik)

1. The player1(master branch) modified the code, and pushed it. But pushing failed, because adversaries modified the code and pushed it earlier.

```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_P
rac_(CO2)/lab/2XB3> git push
To https://github.com/Henry20161020/2XB3.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Henry20161020
/2XB3.git'
hint: Updates were rejected because the remote contains work that yo
u do
hint: not have locally. This is usually caused by another repository
 pushing
hint: to the same ref. You may want to first integrate the remote ch
anges
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for de
tails.
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3>
```

2. The player1(master branch) pulled again, and modified the code.

```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), 381 bytes | 31.00 KiB/s, done.
From https://github.com/Henry20161020/2XB3
   59c5a38..b66290f  master     -> origin/master
Auto-merging lab1/code.py
CONFLICT (content): Merge conflict in lab1/code.py
Automatic merge failed; fix conflicts and then commit the result.
```

Before modify

```
<<<<<<< HEAD
# test1
# students1 = [1111, 2222, 3333]
# students2 = [1111, 2222, 3334]
# groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
# print(are_valid_groups1(students1, groups))
# print(are_valid_groups1(students2, groups))

# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups1(students1, groups1)) # True
print(are_valid_groups1(students2, groups1)) # False
print(are_valid_groups1(students2, groups2)) # False
=======
#noise
# test
students1 = [1111, 2222, 3333]
students2 = [1111, 2222, 3334]
groups = [[11,22,44],[123,432,55],[1111,2222,3333]]

print(are_valid_groups(students2, groups))
>>>>>>> b66290f03b2687ae77e00f8bb5cd3e85525a697a
```

After modify

```
# test1
# students1 = [1111, 2222, 3333]
# students2 = [1111, 2222, 3334]
# groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
# print(are_valid_groups1(students1, groups))
# print(are_valid_groups1(students2, groups))

# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups1(students1, groups1))_# True
print(are_valid_groups1(students2, groups1))_# False
print(are_valid_groups1(students2, groups2))_# False
```

3. The player1(master branch) added, committed, and pushed again, but it also failed. > Maybe one of the adversaries was also on the master branch, so the player one failed to push the file.

```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git add .
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git commit "Resolve the conflict by modifying the code
s after pull"
fatal: cannot do a partial commit during a merge.
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3>
```

```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git push origin master
To https://github.com/Henry20161020/2XB3.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/Henry20161020/2X
B3.git'
hint: Updates were rejected because the tip of your current branch is b
ehind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for detai
ls.
```

4. The player1 changed the branch from master to branch1, repeated the process, and it succeeded.

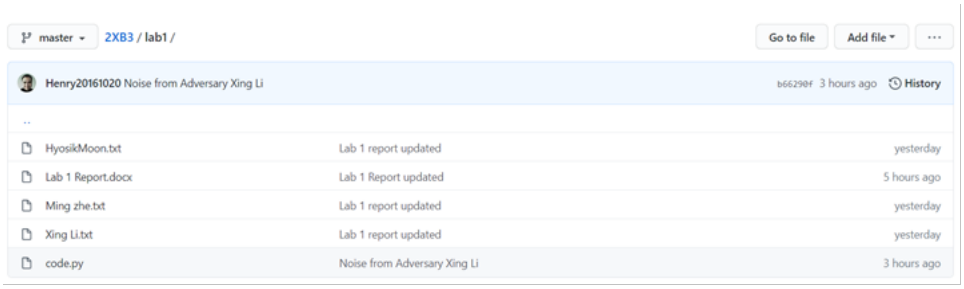
```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git branch
* master
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git checkout -b branch1
Switched to a new branch 'branch1'
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git status
On branch branch1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   lab1/code.py

~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git branch
* branch1
  master
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git push origin branch1
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 2.07 KiB | 1.03 MiB/s, done.
Total 13 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'branch1' on GitHub by visiting:
remote:      https://github.com/Henry20161020/2XB3/pull/new/branch1
remote:
To https://github.com/Henry20161020/2XB3.git
 * [new branch]      branch1 -> branch1
```

5. When the player1(branch1) checked ‘git status’ it was on stage, so he pushed it again with commit ‘Upload 2’.

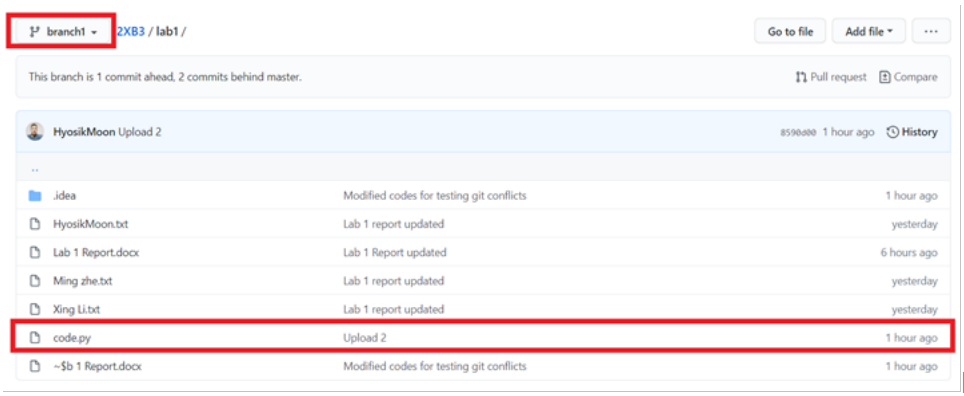
```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git branch
* branch1
  master
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git status
On branch branch1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   lab1/code.py

~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git branch
* branch1
  master
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git add .
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git commit -m "Upload 2"
[branch1 8590d00] Upload 2
 1 file changed, 1 insertion(+), 1 deletion(-)
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git status
On branch branch1
nothing to commit, working tree clean
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git push origin branch1
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 377 bytes | 377.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Henry20161020/2XB3.git
   ce89bff..8590d00  branch1 -> branch1
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(C02)/lab/2XB3> git status
On branch branch1
nothing to commit, working tree clean
```

One of the adversaries' branch(master).

'code.py' was not updated.



But the player1(branch1)'s file was uploaded.



Page Break

6. The player1(branch1 -> master) changed the branch from branch1 to master. When he pulled, CONFLICT appeared.

```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git branch
* branch1
  master
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git checkout master
Switched to branch 'master'
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git branch
  branch1
* master
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git pull
Auto-merging lab1/code.py
CONFLICT (content): Merge conflict in lab1/code.py
Automatic merge failed; fix conflicts and then commit the result.
```

7. When the player1(master) changed the branch, error appeared. It seemed that because he pulled from the master branch, the python file changed. So he modified the file again.

```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git checkout
error: you need to resolve your current index first
lab1/code.py: needs merge
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git checkout branch1
error: you need to resolve your current index first
lab1/code.py: needs merge
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git branch
  branch1
* master
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git merge brach1
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git merge b66290f03b2687ae77e00f8bb5cd3e85525a697a
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git merge branch1
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
```

Before modify

```
<<<<<<< HEAD
# test1
# students1 = [1111, 2222, 3333]
# students2 = [1111, 2222, 3334]
# groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
# print(are_valid_groups1(students1, groups))
# print(are_valid_groups1(students2, groups))

# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups1(students1, groups1)) # True
print(are_valid_groups1(students2, groups1)) # False
print(are_valid_groups1(students2, groups2)) # False
=======
#noise
# test
students1 = [1111, 2222, 3333]
students2 = [1111, 2222, 3334]
groups = [[11,22,44],[123,432,55],[1111,2222,3333]]

print(are_valid_groups(students2, groups))
>>>>>>> b66290f03b2687ae77e00f8bb5cd3e85525a697a
```

After modify

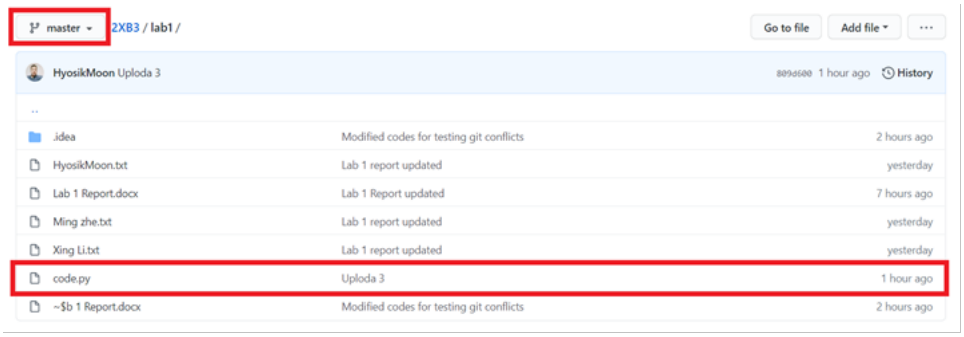
```
# test1
# students1 = [1111, 2222, 3333]
# students2 = [1111, 2222, 3334]
# groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
# print(are_valid_groups1(students1, groups))
# print(are_valid_groups1(students2, groups))

# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups1(students1, groups1))# True
print(are_valid_groups1(students2, groups1))# False
print(are_valid_groups1(students2, groups2))# False
```

Page Break

8. After changing the code again, the player1 added, committed, and pushed the file again. And It worked.

```
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git merge b66290f03b2687ae77e00f8bb5cd3e85525a697a
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   lab1/code.py

no changes added to commit (use "git add" and/or "git commit -a")
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git add .
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git commit -m "Uploda 3"
[master 809d600] Uploda 3
~/Desktop/Study/Mcmaster/2021 Winter/2XB3 CS_Pract_&_Exp_Theory_To_Prac
_(CO2)/lab/2XB3> git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 404 bytes | 404.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Henry20161020/2XB3.git
   b66290f..809d600  master -> master
```

File updated(Upload3) !.



* Discussion

1. Why didn't it work when the player1(master branch) tried to push at first? Maybe one of the adversaries had been online as a master branch too. After an hour when the adversary was logged out automatically, the player1(master branch) can push the file.

Answer by Xing: There could be some updates from other collaborators between two of your pulls when you are resolving merge conflicts. This might happen if the conflict resolving took you a long time.

2. In order to avoid conflicts, modify codes at a different branch. (* Don't modify it on the master branch.)

Answer by Xing: I think it's a good habit. The merge will happen between the branch and the master.

3. When conflicts happen, pull the code again, modify, and push it. (* or Merge other branches when they are using different branches)

Answer by Xing: When encountering conflicts during pulling, git will automatically merge. If automatic merge fails, it will tell us which files cause the failure. Then we can modify, commit and push.

Second Round (Player = Xing Li, Adversary = Mingzhe Wang, Writer = Xing Li)

After modification, I pulled the origin. As expected, there is merge conflict in lab1/code.py because of adversary.

```
MacBooks-MBP:2XB3 macbook$ git commit -m "player Xing Li finishes code modification."
[master a920022] player Xing Li finishes code modification.
 2 files changed, 18 insertions(+), 37 deletions(-)
MacBooks-MBP:2XB3 macbook$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/Henry20161020/2XB3
   c8c6507..2a85051  master     -> origin/master
Auto-merging lab1/code.py
CONFLICT (content): Merge conflict in lab1/code.py
Automatic merge failed; fix conflicts and then commit the result.
```

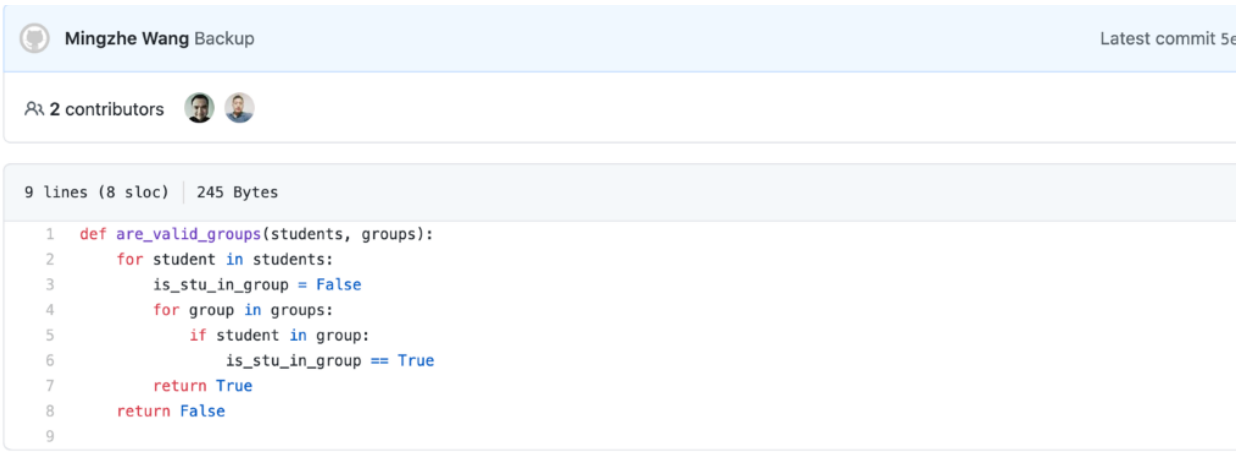
I need to clean those messages caused by adversary

```
# Created by Hyosik
def are_valid_groups1(students, groups):
    output = False
    student_num = len(students)
    true_num = 0
    for group in groups:
<<<<<<< HEAD
        for student in students:
            if student in group:
                true_num += 1
        if true_num == student_num:
            output = True
            break
=======
        if len(group) == 2 or len(group) == 3:
            for student in students:
                if student in group:
                    true_num += 1
            if true_num == student_num:
                output = True
                break
####################annoyingannoyingannoyingannoyingannoyingannoying######
        else:
            output = False
    #Check if a student occurs just once in the groups
    for student in students:
        if groups_onelist.count(student) == 1:
            output = True
            ####################annoyingannoyingannoyingannoyingannoyingannoying######
        else:
            output = False
>>>>>>> 2a85051bcff41fae7a147c05e1a6b6c3c7690d15
    return output

<<<<<<< HEAD
# test
students1 = [1111, 2222, 3333]
students2 = [1111, 2222, 3334]
groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
```

I committed the merge, and I pulled again to verify there was no change in the origin when I was resolving the conflicts. Then I pushed my update.

Third Round (Player = Mingzhe Wang, Adversary = Hyosik, Writer = Mingzhe Wang)

1. In this round, I experimented “git reset”. Before the game began, I pulled from the remote server and made sure my repositories are updated.




```
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git pull
Already up to date.
```

2.Then I started to revise my code in my local machine.

```python
def are_valid_groups(students, groups):
    output = False
    student_num = len(students)
    true_num = 0
    groups_onelist = [student_number for group in groups for student_number in group]
    for group in groups:
        for student in students:
            if student in group:
                true_num += 1
        if true_num == student_num:
            output = True
            break

        if len(group) == 2 or len(group) == 3:
            for student in students:
                if student in group:
                    true_num += 1
            if true_num == student_num:
                output = True
                break
        else:
            output = False
    #Check if a student occurs just once in the groups
    for student in students:
        if groups_onelist.count(student) == 1:
            output = True
        else:
            output = False
    return output

# test
#students1 = [1111, 2222, 3333]
#students2 = [1111, 2222, 3334]
#groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
#print(are_valid_groups(students2, groups))
# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups(students1, groups1)) # True
print(are_valid_groups(students2, groups1)) # False
print(are_valid_groups(students2, groups2)) # False
```

3. At the meantime, the adversary was making some noises. I noticed that the code.py file in the server at that time was changed to the following:

```python
def are_valid_groups2222222222222222222(students, groups):
    for student in students:
        is_stu_in_group = False
        for group in groups:
            if student in group:
                is_stu_in_group == True
        return 2222222222222222222True
    return False

```

4. Then I tried to push my revised code, but I got some warnings of rejecting the push.

```
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git add code.py
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git commit -m "update my implementa
tion"
[master e702767] update my implementation
 1 file changed, 41 insertions(+), 6 deletions(-)
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push
To https://github.com/Henry20161020/2XB3.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Henry20161020/2XB3.git
'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

5. To resolve this conflict, I pulled from the server, and noticed that there are some conflicts happening in the file.

```
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/Henry20161020/2XB3
   5e9becd..617d25d  master     -> origin/master
Auto-merging lab1/code.py
CONFLICT (content): Merge conflict in lab1/code.py
Automatic merge failed; fix conflicts and then commit the result.
```

6. Then I went to the file and checked the conflicts.

```
<<<<<<< HEAD
def are_valid_groups(students, groups):
    output = False
    student_num = len(students)
    true_num = 0
    groups_onelist = [student_number for group in groups for student_number in
        group]
    for group in groups:
        for student in students:
            if student in group:
                true_num += 1
        if true_num == student_num:
            output = True
            break

        if len(group) == 2 or len(group) == 3:
            for student in students:
                if student in group:
                    true_num += 1
            if true_num == student_num:
                output = True
                break
        else:
            output = False
    #Check if a student occurs just once in the groups
    for student in students:
        if groups_onelist.count(student) == 1:
            output = True
        else:
            output = False
    return output
```

```
# test
#students1 = [1111, 2222, 3333]
#students2 = [1111, 2222, 3334]
#groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
#print(are_valid_groups(students2, groups))
# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups(students1, groups1)) # True
print(are_valid_groups(students2, groups1)) # False
print(are_valid_groups(students2, groups2)) # False
=======
def are_valid_groups22222222222222222(students, groups):
    for student in students:
        is_stu_in_group = False
        for group in groups:
            if student in group:
                is_stu_in_group == True
        return 22222222222222222222True
    return False
>>>>>>> 617d25dce780895c94e76832ca5243a4acca91a8
```

7. I selected the part that I want to be resolved and then remove the noising part.

```python
def are_valid_groups(students, groups):
    output = False
    student_num = len(students)
    true_num = 0
    groups_onelist = [student_number for group in groups for student_number in
        group]
    for group in groups:
        for student in students:
            if student in group:
                true_num += 1
        if true_num == student_num:
            output = True
            break

        if len(group) == 2 or len(group) == 3:
            for student in students:
                if student in group:
                    true_num += 1
            if true_num == student_num:
                output = True
                break
        else:
            output = False
    #Check if a student occurs just once in the groups
    for student in students:
        if groups_onelist.count(student) == 1:
            output = True
        else:
            output = False
    return output

# test
#students1 = [1111, 2222, 3333]
#students2 = [1111, 2222, 3334]
#groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
#print(are_valid_groups(students2, groups))
# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups(students1, groups1)) # True
print(are_valid_groups(students2, groups1)) # False
print(are_valid_groups(students2, groups2)) # False
```

8. After finishing this, I tried to push again. This time, there's no warning showed.

```
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git add code.py
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git commit -m "Resolve the noises"
[master 6309e20] Resolve the noises
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.10 KiB | 1.10 MiB/s, done.
Total 8 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/Henry20161020/2XB3.git
   617d25d..6309e20  master -> master
```

9. After my push was finished, the code.py in the server was showed as the following:

```python
def are_valid_groups(students, groups):
    output = False
    student_num = len(students)
    true_num = 0
    groups_onelist = [student_number for group in groups for student_number in group]
    for group in groups:
        for student in students:
            if student in group:
                true_num += 1
        if true_num == student_num:
            output = True
            break

        if len(group) == 2 or len(group) == 3:
            for student in students:
                if student in group:
                    true_num += 1
            if true_num == student_num:
                output = True
                break
        else:
            output = False
    #Check if a student occurs just once in the groups
    for student in students:
        if groups_onelist.count(student) == 1:
            output = True
        else:
            output = False
    return output

# test
#students1 = [1111, 2222, 3333]
#students2 = [1111, 2222, 3334]
#groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
```

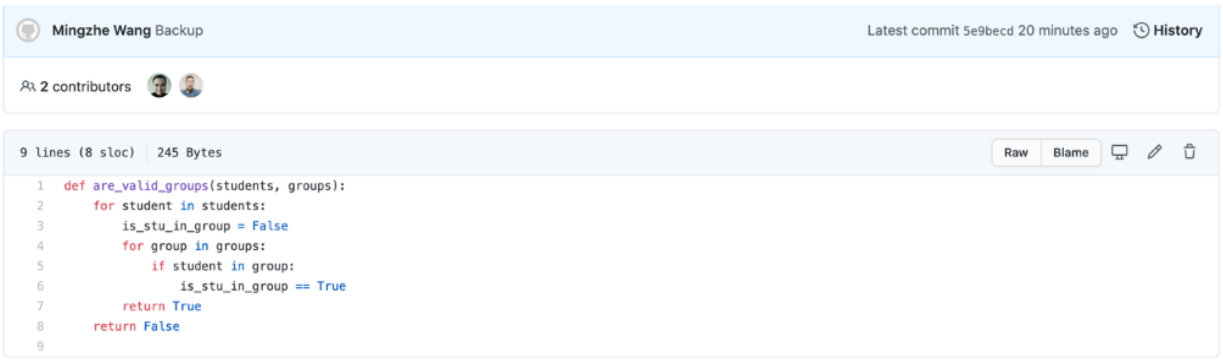
```
                    true_num += 1
            if true_num == student_num:
                output = True
                break
        else:
            output = False
    #Check if a student occurs just once in the groups
    for student in students:
        if groups_onelist.count(student) == 1:
            output = True
        else:
            output = False
    return output

# test
#students1 = [1111, 2222, 3333]
#students2 = [1111, 2222, 3334]
#groups = [[11,22,44],[123,432,55],[1111,2222,3333]]
#print(are_valid_groups(students2, groups))
# test2
students1 = ['1111', '2222', '3333']
students2 = ['1111', '2222', '3334']
groups1 = [['11','22','44'],['123','432','55'],['1111','2222','3333']]
groups2 = [['11','22','44'],['123','3333','55'],['1111','2222','3333']]
print(are_valid_groups(students1, groups1)) # True
print(are_valid_groups(students2, groups1)) # False
print(are_valid_groups(students2, groups2)) # False
```

10. At last, we used command “git reset –hard [commit number]” and “git push -f” to reset the commit to let the remote server being prepared for the next round of game.

```
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git log --oneline
6309e20 (HEAD -> master, origin/master, origin/HEAD) Resolve the noises
e702767 update my implementation
617d25d Noise from Hyosik
5e9becd Backup
fc7c948 Game Beginning
ccc6624 This is the beginning of the game
b0feff5 reset
0206a0a against adversary of MZ Merge branch 'master' of https://github.com/H
enry20161020/2XB3
4a272c2 my code is back
b14a867 make some noises
00693b4 my code removed
2fd5c7c my code.py inserted
```

```
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git -reset --hard 5e9becd
unknown option: -reset
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare
]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git reset --hard 5e9becd
HEAD is now at 5e9becd Backup
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push
To https://github.com/Henry20161020/2XB3.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/Henry20161020/2XB3.git
'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push -f
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/Henry20161020/2XB3.git
 + 6309e20...5e9becd master -> master (forced update)
(base) Mingzhes-MacBook-Pro:lab1 kidsama$
```

The code.py file on the server was reset to the beginning status as the following:



11.This is the whole record of my terminal during that time.

(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git pull
Already up to date.
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git add code.py
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git commit -m "update my implementation"
[master e702767] update my implementation
 1 file changed, 41 insertions(+), 6 deletions(-)
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push
To https://github.com/Henry20161020/2XB3.git
 ! [rejected]   master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Henry20161020/2XB3.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/Henry20161020/2XB3

```
   5e9becd..617d25d  master     -> origin/master
Auto-merging lab1/code.py
CONFLICT (content): Merge conflict in lab1/code.py
Automatic merge failed; fix conflicts and then commit the result.
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git add code.py
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git commit -m "Resolve the noises"
[master 6309e20] Resolve the noises
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.10 KiB | 1.10 MiB/s, done.
Total 8 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/Henry20161020/2XB3.git
   617d25d..6309e20  master -> master
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git log --oneline
6309e20 (HEAD -> master, origin/master, origin/HEAD) Resolve the noises
e702767 update my implementation
617d25d Noise from Hyosik
5e9becd Backup
fc7c948 Game Beginning
ccc6624 This is the beginning of the game
b0feff5 reset
0206a0a against adversary of MZ Merge branch 'master' of
https://github.com/Henry20161020/2XB3
4a272c2 my code is back
b14a867 make some noises
00693b4 my code removed
2fd5c7c my code.py inserted
31c75d2 received new code
5bbf479 resolve conflict_Hyosik
6f45251 Hyosik's file2
```

2e102e9 Mingzhe's code
fa9986c Reset
3b2dfe7 Mingzhe's code
8d0ea6a everything reset
d77970d Lab 1 report updated
69bc25d Adversary message cleaned and merged
a920022 player Xing Li finishes code modification.
2a85051 make some noises
c8c6507 Lab 1 Report Updated
2ac1143 Merge branch 'branch1'
809d600 Uploda 3
8590d00 (**origin/branch1**) Upload 2
ce89bff Modified codes for testing git conflicts
b66290f Noise from Adversary Xing Li
59c5a38 Lab 1 Report updated
03c1454 Merge all three
6eeef9f implement are_valid_groups
25068d2 Lab 1 Report updated
29bcb8d Hyosik and Henry's code.py merged. Lab 1 Report updated
fc4072b code.py modified by Xing Li
a0004fe Add code.py file, Hyosik
4e8af91 Delete ~$b 1 Report.docx
f36a4ec Lab 1 folder removed
bf02d6e Lab 1 report updated
9c6d23a Hyosik and Mingzhe txt merged
694fbb6 Upload Hyosik's file
d2dc22a empty code.py posted
765049c This is Mingzhe's txt
84776e1 Lab contract modified with task complete message required
42e2641 Xing Li txt posted
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git -reset --hard 5e9becd
unknown option: -reset
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]

```
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git reset --hard 5e9becd
HEAD is now at 5e9becd Backup
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push
To https://github.com/Henry20161020/2XB3.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/Henry20161020/2XB3.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
(base) Mingzhes-MacBook-Pro:lab1 kidsama$ git push -f
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/Henry20161020/2XB3.git
 + 6309e20...5e9becd master -> master (forced update)
(base) Mingzhes-MacBook-Pro:lab1 kidsama$
```

```
MacBooks-MBP:2XB3 macbook$ git add lab1/code.py
MacBooks-MBP:2XB3 macbook$ git commit -m "Adversary message cleaned and merged"
[master 69bc25d] Adversary message cleaned and merged
MacBooks-MBP:2XB3 macbook$ git pull
Already up to date.
MacBooks-MBP:2XB3 macbook$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

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

        modified:   lab1/Lab 1 Report.docx

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

        .DS_Store
        lab1/.DS_Store

no changes added to commit (use "git add" and/or "git commit -a")
MacBooks-MBP:2XB3 macbook$ git push
Enumerating objects: 17, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 10.41 KiB | 10.41 MiB/s, done.
Total 9 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), completed with 3 local objects.
To https://github.com/Henry20161020/2XB3.git
   2a85051..69bc25d  master -> master
```

What is Learned about Conflicts?

Throughout the lab, we encountered different conflicts requiring merge. The easiest one to handle is new file either in the origin or in the local environment. They can be automatically merged. The one requiring more efforts are file name changes or folder name changes. Both git add and git rm are required. The trickiest one is the conflicts in one file. This kind of conflict cannot be automatically merged. Git pull/merge, editing in a text editor and git commit are required to resolve the conflicts.

To avoid conflict issues, there could be some “good” habits. First, commit more often. As long as there is no one touching the origin repository between your pull and push, there will be no conflicts. Second, when git push failed, pull the origin first, resolve the conflict and commit to merge. This is most straightforward way to deal with conflicts.