CPE 327 SOFTWARE ENGINEERING, KMUTT

CONFIGURATION MANAGEMENT LAB SHEET.

DUE ON Tue 28 Nov at 11.59 PM.

THIS IS NOT A GROUP ASSIGNMENT, BUT MUST BE DONE WITH 1 PARTNER OR DONE ALONE. IF DONE ALONE YOU NEED 2 GIT ACCOUNTS.

In this year's lab, we will be using GitHub instead of a CVS tool. Both systems can track changes to the files in your project and inform you of conflicts created when more than one developer make changes to the same file. Each developer works in an individual directory and then merges the work from each after the work is complete. CVS does not maintain multiple versions of source code files but keeps a single copy and records of all of the changes that are made However, Git works by maintaining snapshots that are entire files, not just initial files + changes.

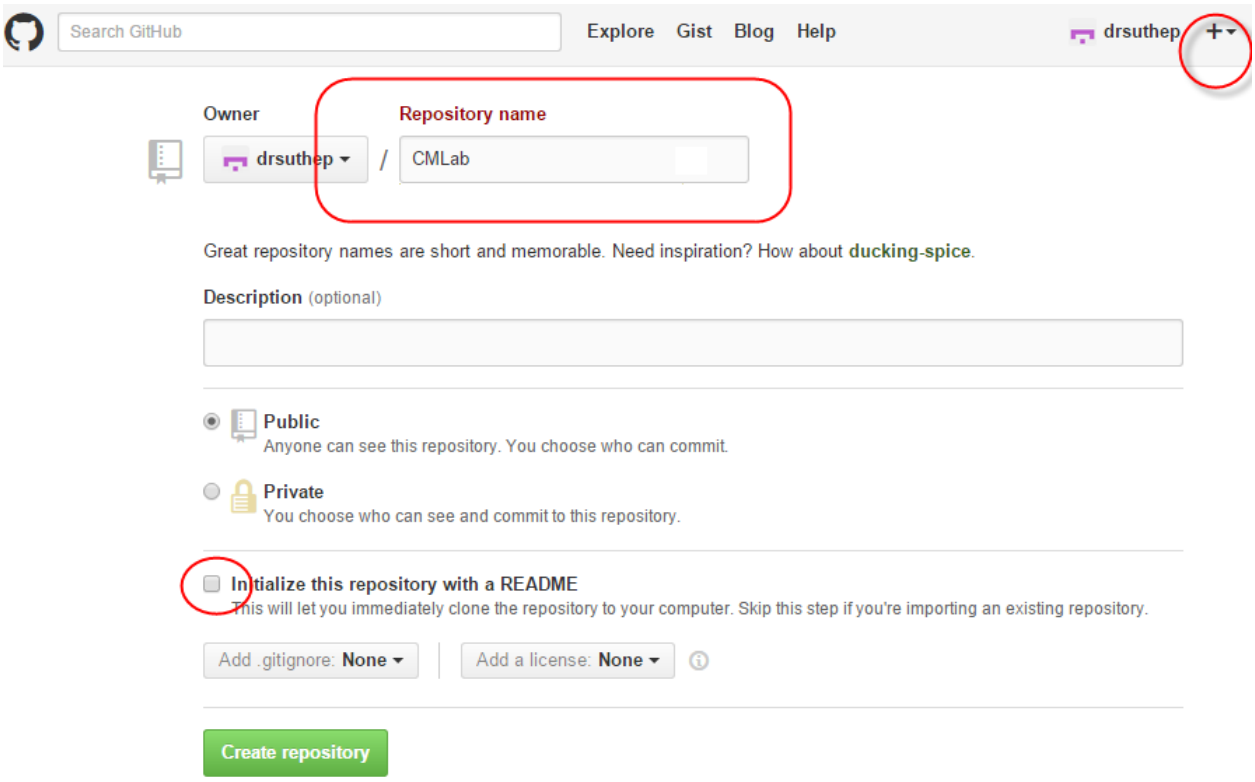
Follow the instructions below to complete this lab.

Installation and preparation

1. Sign up for GitHub account at www.github.com
2. Download and install Git software for PC or Mac. GitHub won't work on your local computer if you don't install Git. Install Git for Windows, Mac or Linux. (see http://git-scm.com/downloads)
3. Run git commands on the Git Bash shell (terminal).

My Example Workflow Steps

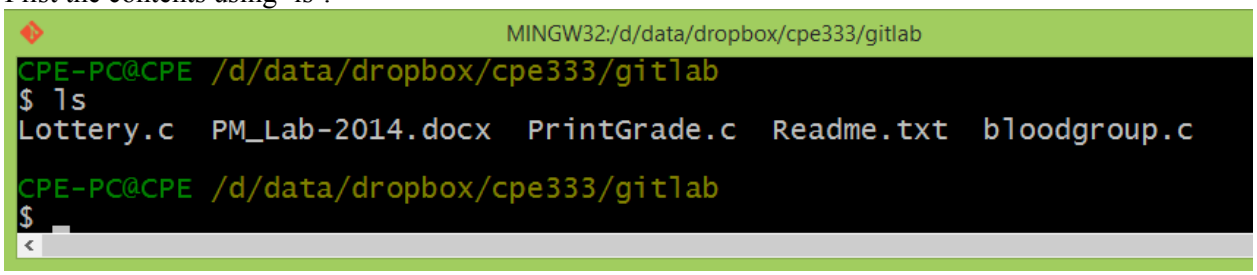
1. Create a blank repository in GitHub with no readme file. Print hub status.



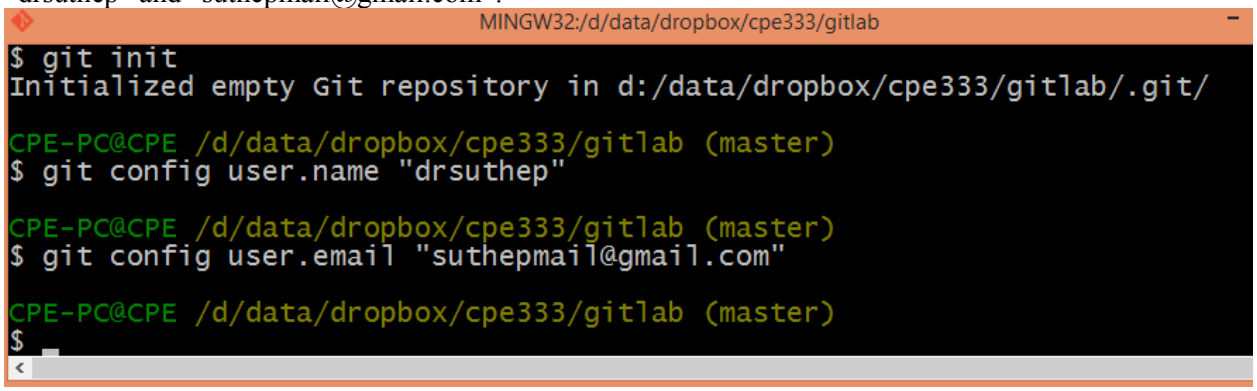
2. Assume you already have 1 Readme.txt file, 2 C programs (Program #1 and Program #2), and 1 MS Word document in your local PC. Make a git repository on that PC's folder and then push it to the GitHub repository.

- I have my files waiting in d:/data/dropbox/cpe333/gitlab, so I go there first using ‘cd’ to change directory.

- I list the contents using ‘ls’:



- I then make this a .git repository using “git init”. Then I set the user name and email for this repository to “drsuthep” and “suthepmail@gmail.com”:



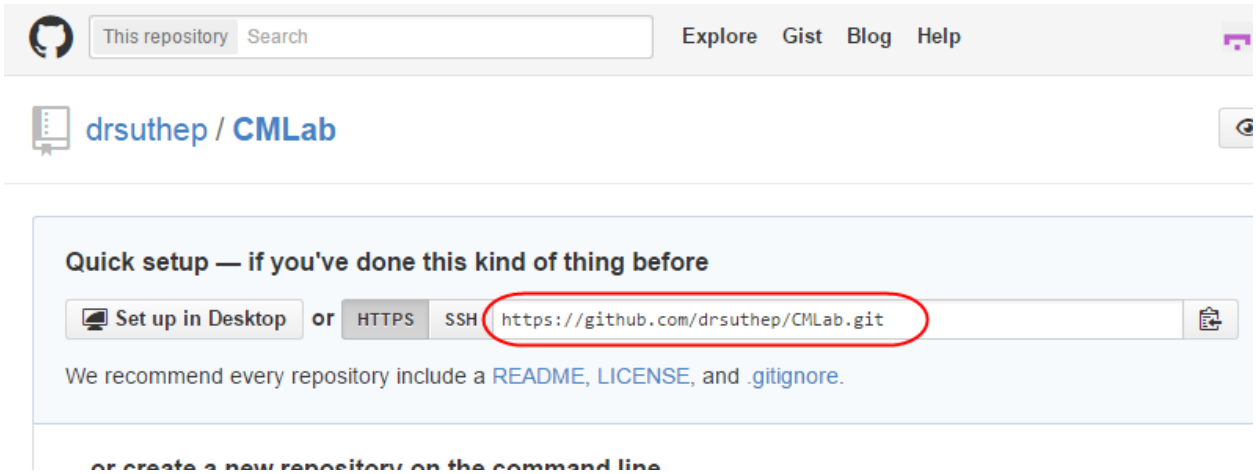
- I then add all 4 files to the git repository using add. And then commit. Then push it to GitHub. Upon adding they become staged.

```
CPE-PC@CPE /d/data/dropbox/cpe333/gitlab (master)
$ git status
On branch master

Initial commit

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

        Lottery.c
        PM_Lab-2014.docx
        PrintGrade.c
        Readme.txt
        bloodgroup.c

nothing added to commit but untracked files present (use "git add" to track)

CPE-PC@CPE /d/data/dropbox/cpe333/gitlab (master)
$ git add *

CPE-PC@CPE /d/data/dropbox/cpe333/gitlab (master)
$ git status
On branch master

Initial commit

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

        new file:   Lottery.c
        new file:   PM_Lab-2014.docx
        new file:   PrintGrade.c
        new file:   Readme.txt
        new file:   bloodgroup.c
```

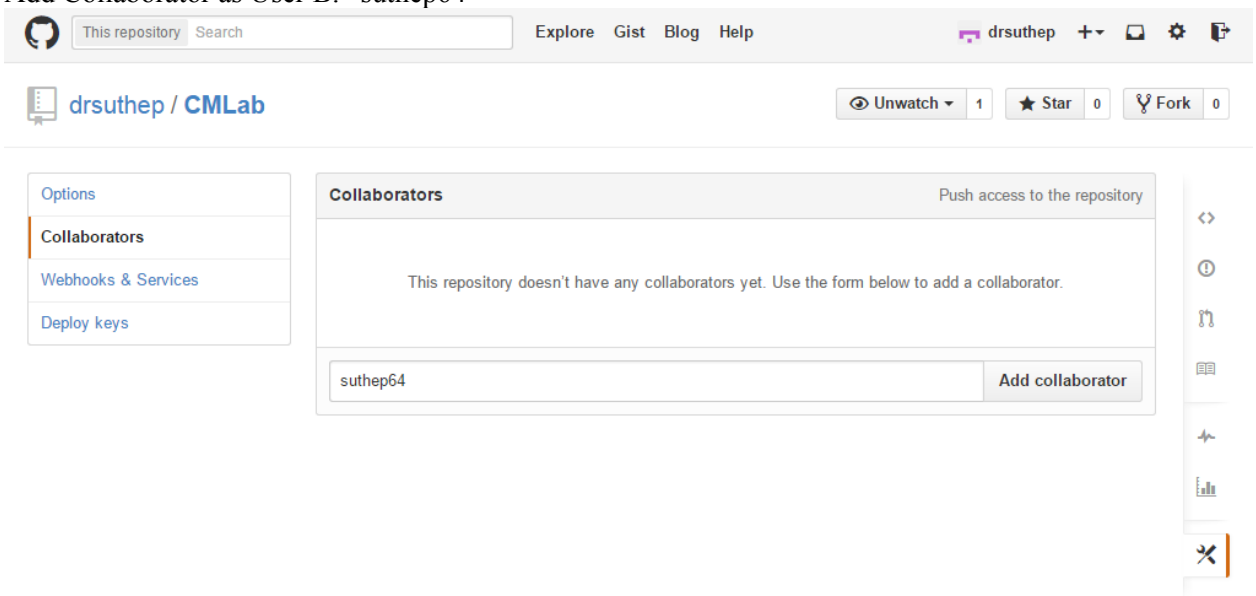
- Now we commit these changes with a note as “First copy of 5 files”

```
MINGW32:/d/data/dropbox/cpe333/gitla
CPE-PC@CPE /d/data/dropbox/cpe333/gitlab (master)
$ git commit -m "First copy of 5 files"
[master (root-commit) 335fd7a] First copy of 5 files
 5 files changed, 118 insertions(+)
 create mode 100644 Lottery.c
 create mode 100644 PM_Lab-2014.docx
 create mode 100644 PrintGrade.c
 create mode 100644 Readme.txt
 create mode 100644 bloodgroup.c

CPE-PC@CPE /d/data/dropbox/cpe333/gitlab (master)
$ git status
On branch master
nothing to commit, working directory clean

CPE-PC@CPE /d/data/dropbox/cpe333/gitlab (master)
$
```

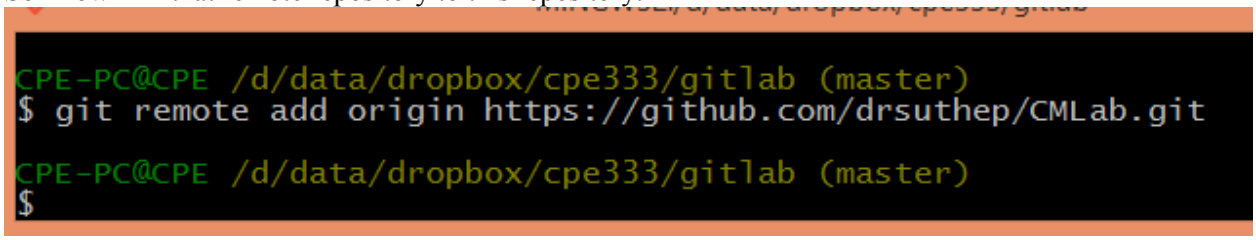
- The GitHub repository is located at https://github.com/drsuthep/CMLab.git as can be seen here:



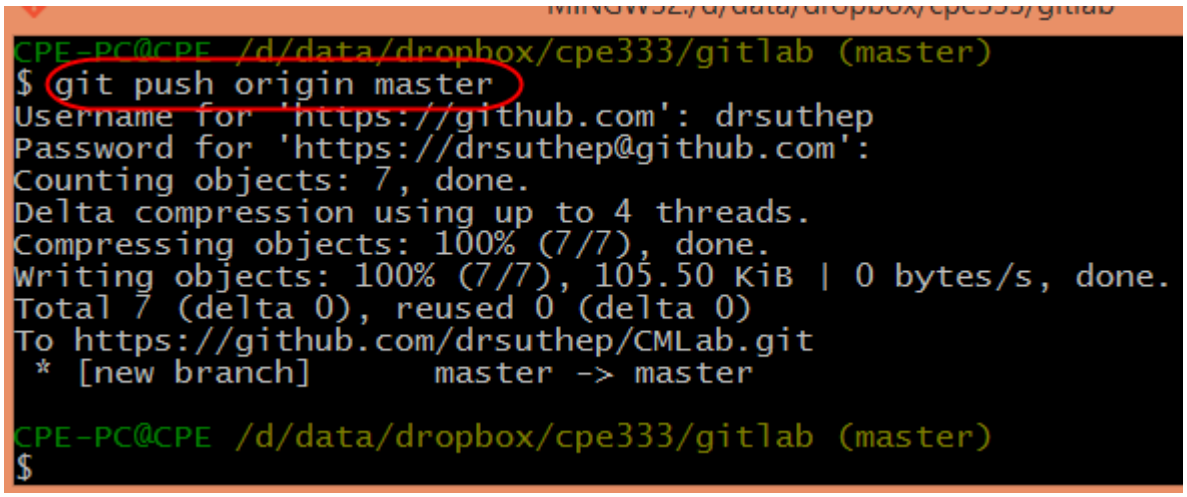
- Add Collaborator as User B: "suthep64"



- So I now link that remote repository to this repository.

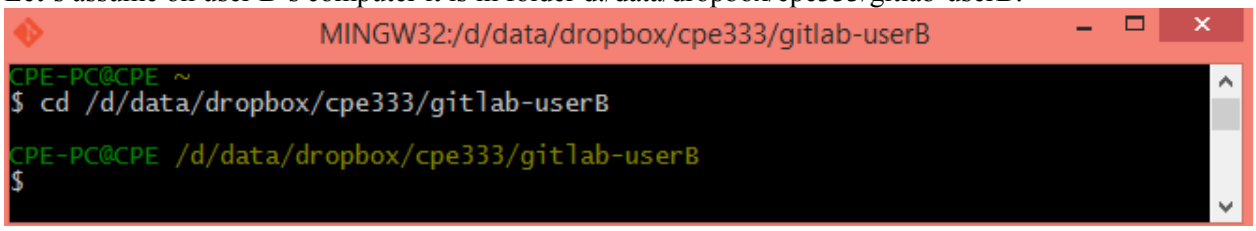


- Now I push my local folder (called origin master) to the GitHub repository.

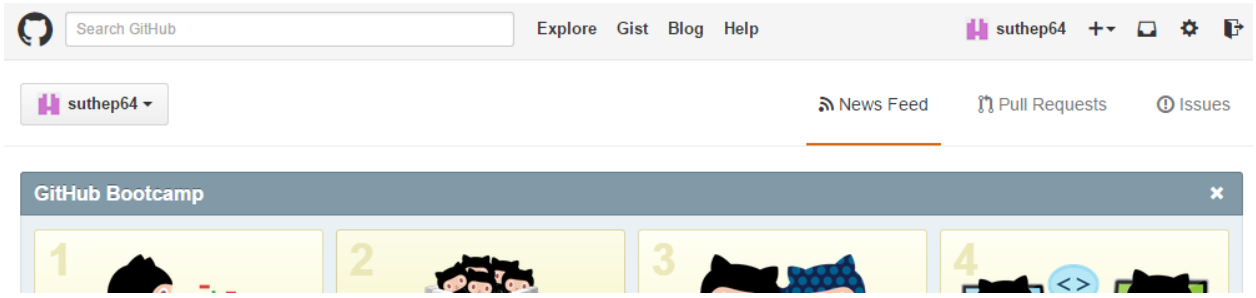


3. User B creates a copy of that repository locally in another folder of his/her own computer. Print local status Users B.

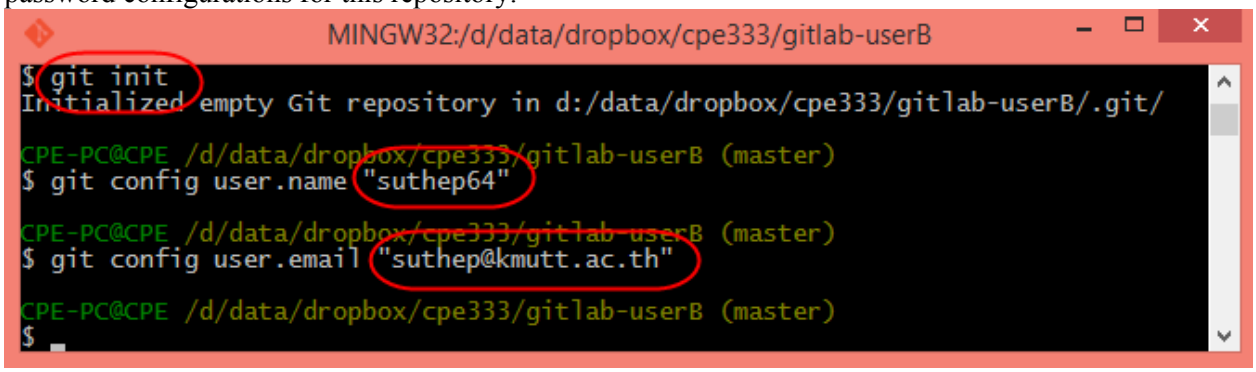
- Let's assume on user B's computer it is in folder d:/data/dropbox/cpe333/gitlab-userB:



- I will use the second user or user B as "suthep64" (I used 2 email accounts to simulate 2 users for myself). This account has email of "suthep@kmutt.ac.th", my other email account.



- I initialize the local repository at the current folder to create a .git repository and then set the user name and password configurations for this repository:



- Now I inform the local directory where the source GitHub is and then pull that

```
MINGW32:/d/data/dropbox/cpe333/gitlab-userB
$ git remote add origin https://github.com/drsuthep/CMLab.git

CPE-PC@CPE /d/data/dropbox/cpe333/gitlab-userB (master)
$ git pull origin master
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
From https://github.com/drsuthep/CMLab
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master

CPE-PC@CPE /d/data/dropbox/cpe333/gitlab-userB (master)
$
```

4. User A change 1 function in Program #1. Print local status User A.

- Before Change file "bloodgroup.c":

File Edit Format View Help

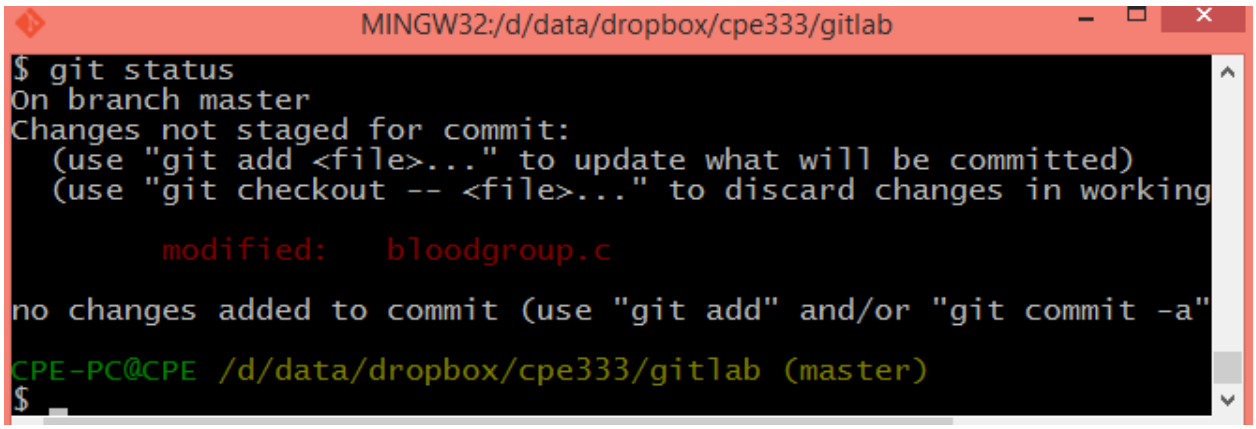
```
#include <stdio.h>
#include <string.h>
void main()
{ char Name[40], G[5], answer[10];
  int i;
  do
  {
    printf("What is your Name? ");
    scanf("%s", Name);
```

- After Change file "bloodgroup.c":

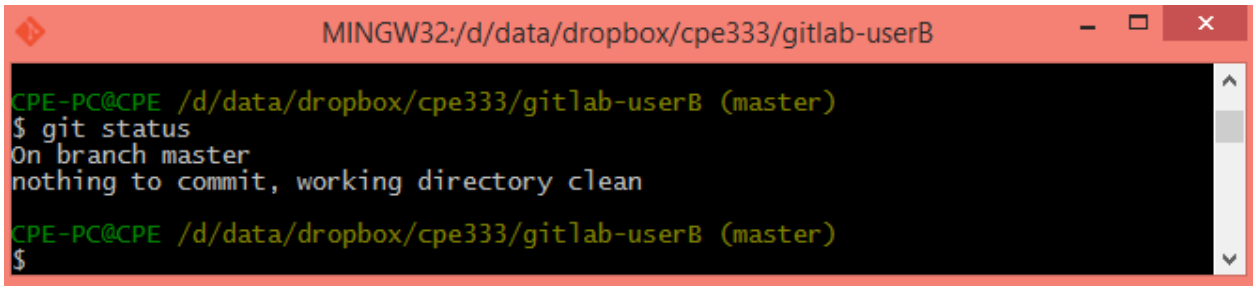
File Edit Format View Help

```
#include <stdio.h>
#include <string.h>
void main()
{ char Name[50], G[5], answer[10];
  int i;
  do
  {
    printf("What is your name? ");
    scanf("%s", Name);
```

- User A's Git Status:

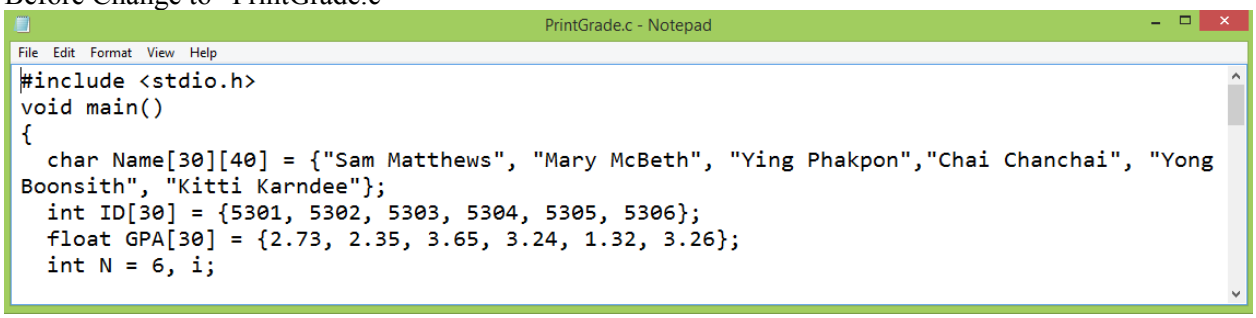


- User B's Git Status:

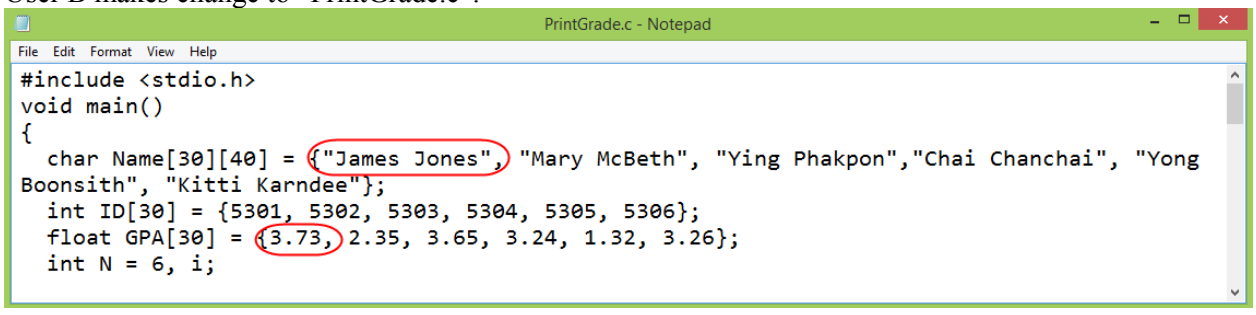


5. User B change 1 function in Program #2. Print local status User B.

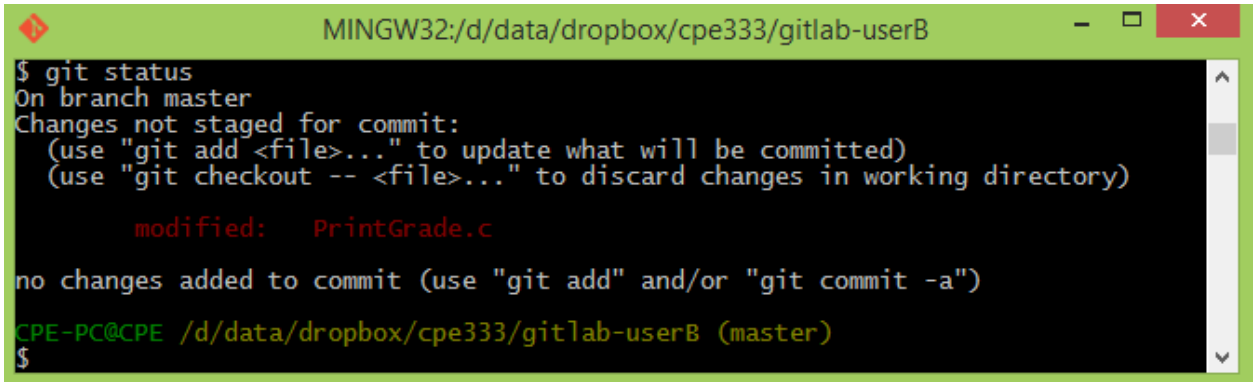
- Before Change to "PrintGrade.c"



- User B makes change to "PrintGrade.c":

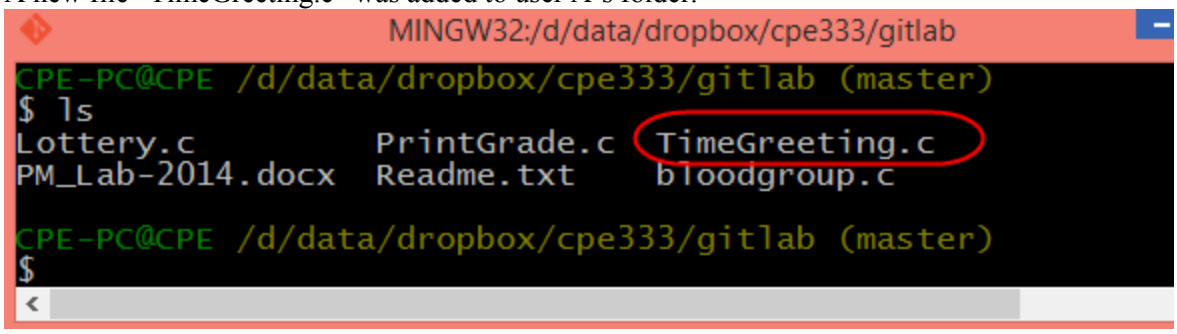


- User B Status:

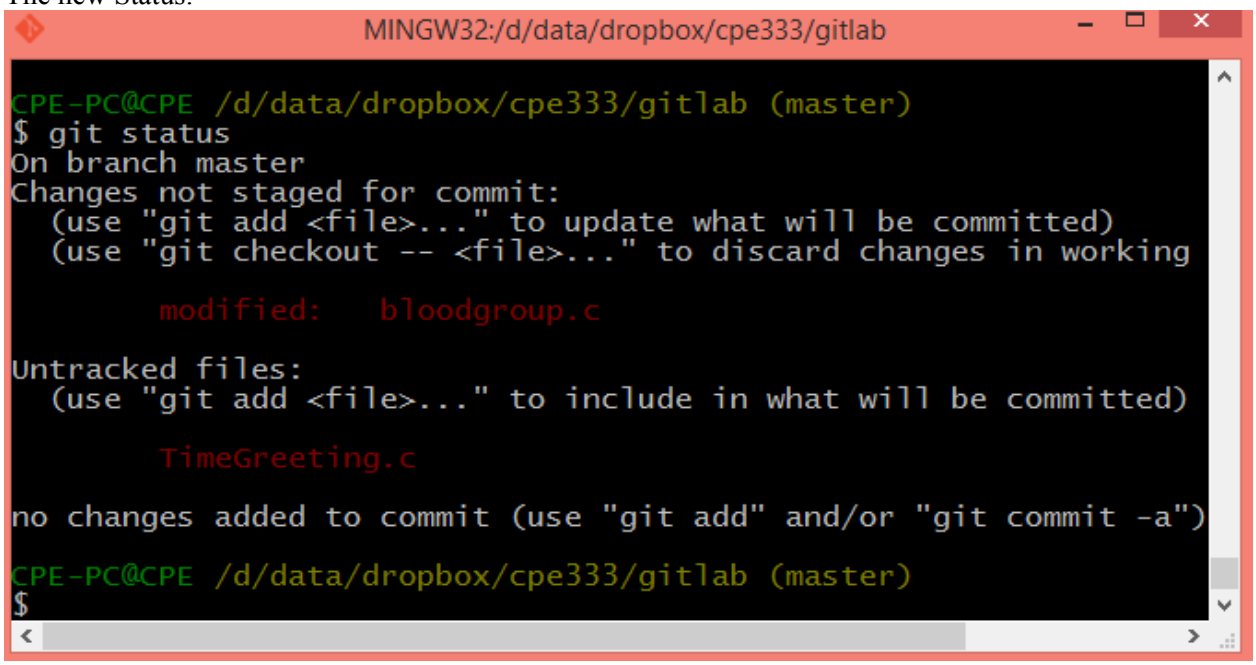


6. User A create a new C program Program #3. Do not put into Git yet. Print local status User A.

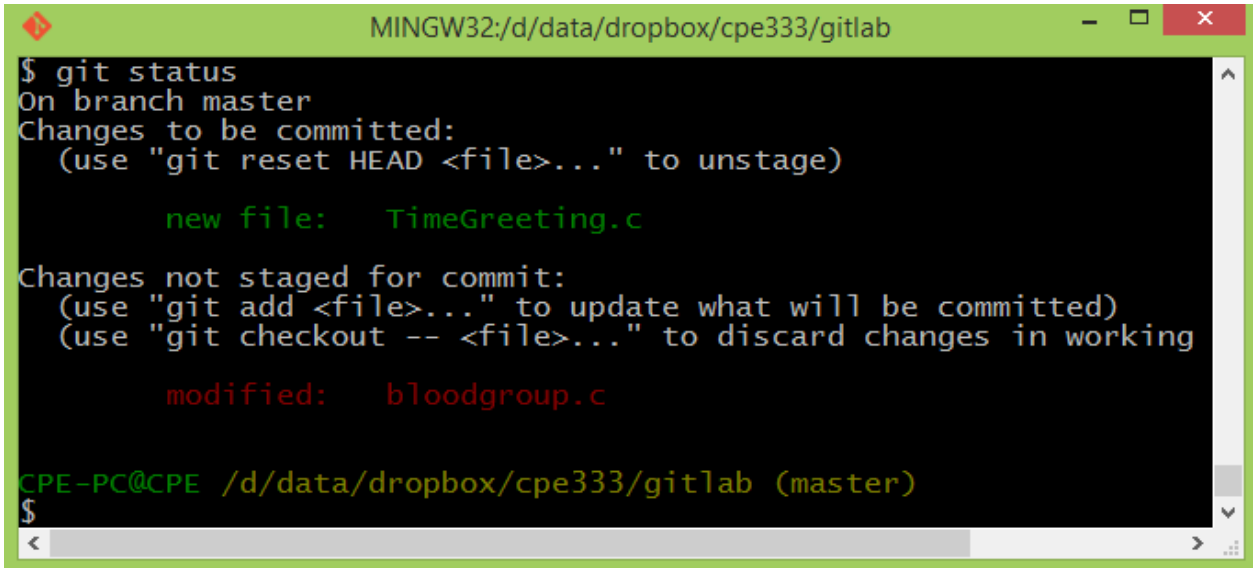
- A new file "TimeGreeting.c" was added to user A's folder.



- The new Status:

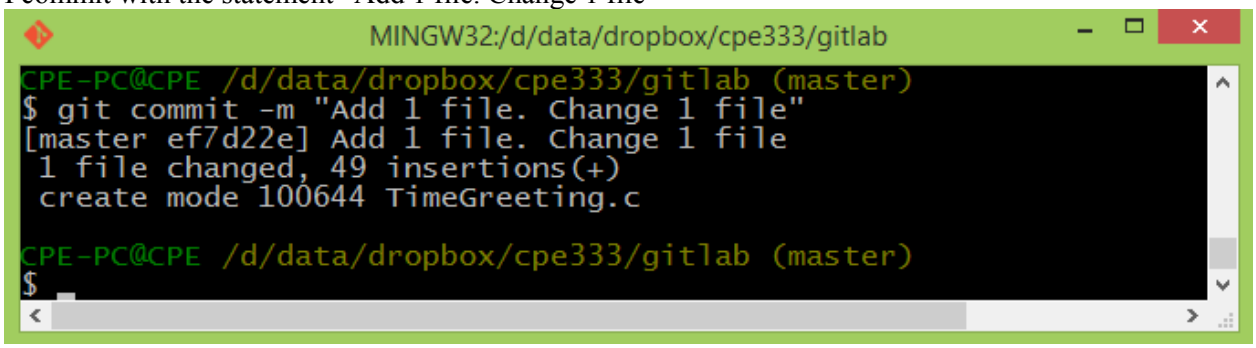


7. User A adds Program #3 to Git control (git add). Print local status User A.
   $ git add TimeGreeting.c //Please add this missing line.

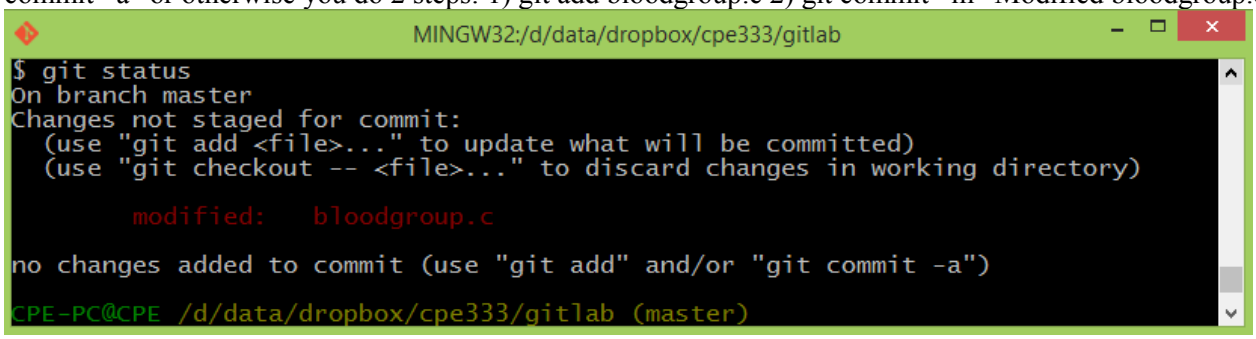


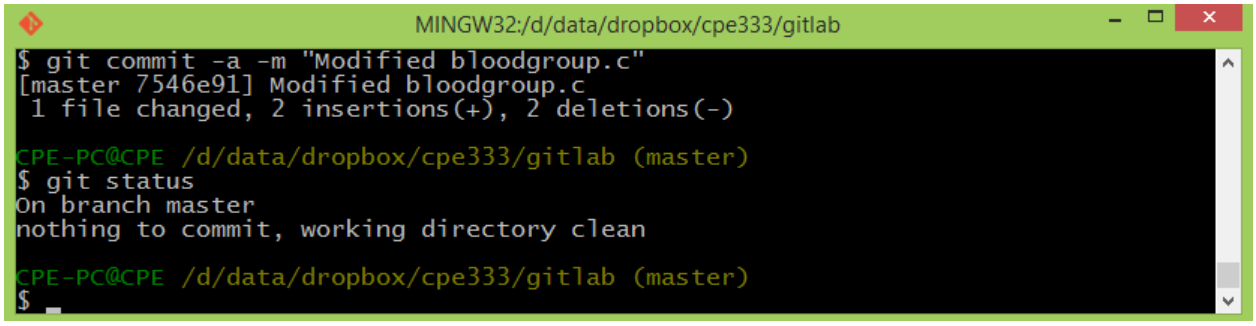
8. User A commits changes (note that changed are Program #1 and Program #3). Print local status User A.

- I commit with the statement "Add 1 file. Change 1 file"

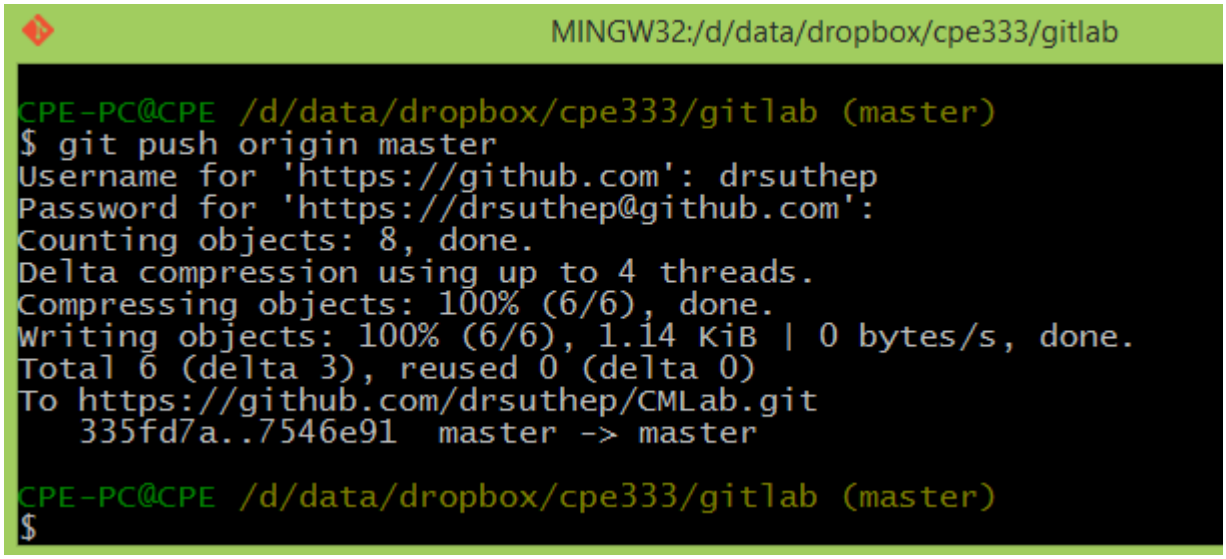


- With git status I notice I forgot to add and commit 1 file. You can add and commit at once by using "git commit –a" or otherwise you do 2 steps: 1) git add bloodgroup.c 2) git commit –m "Modified bloodgroup.c".

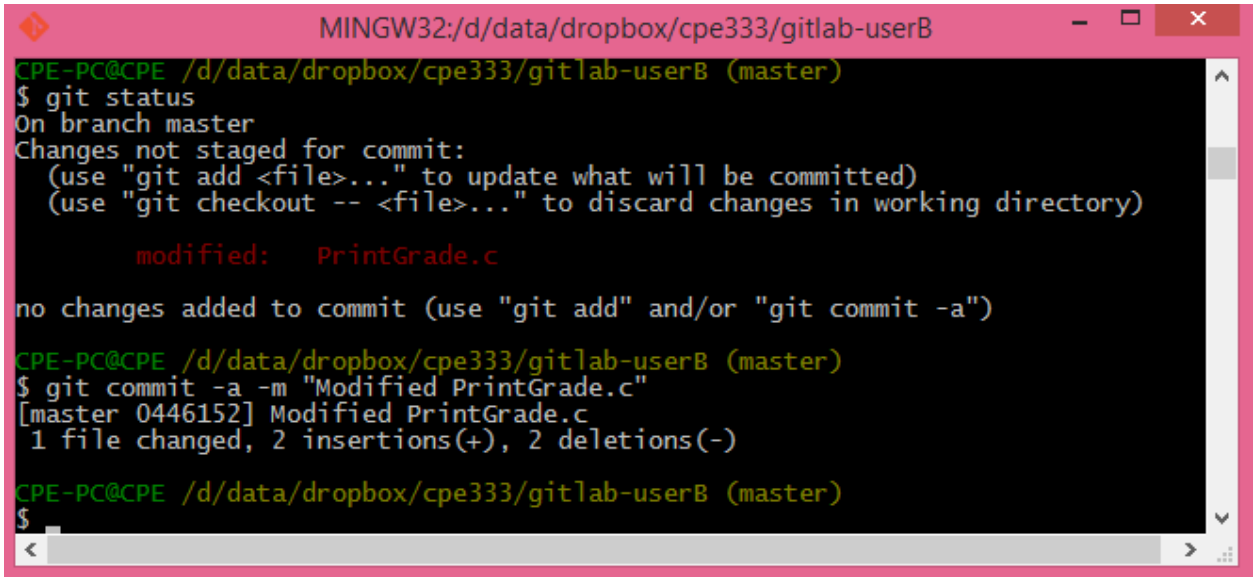




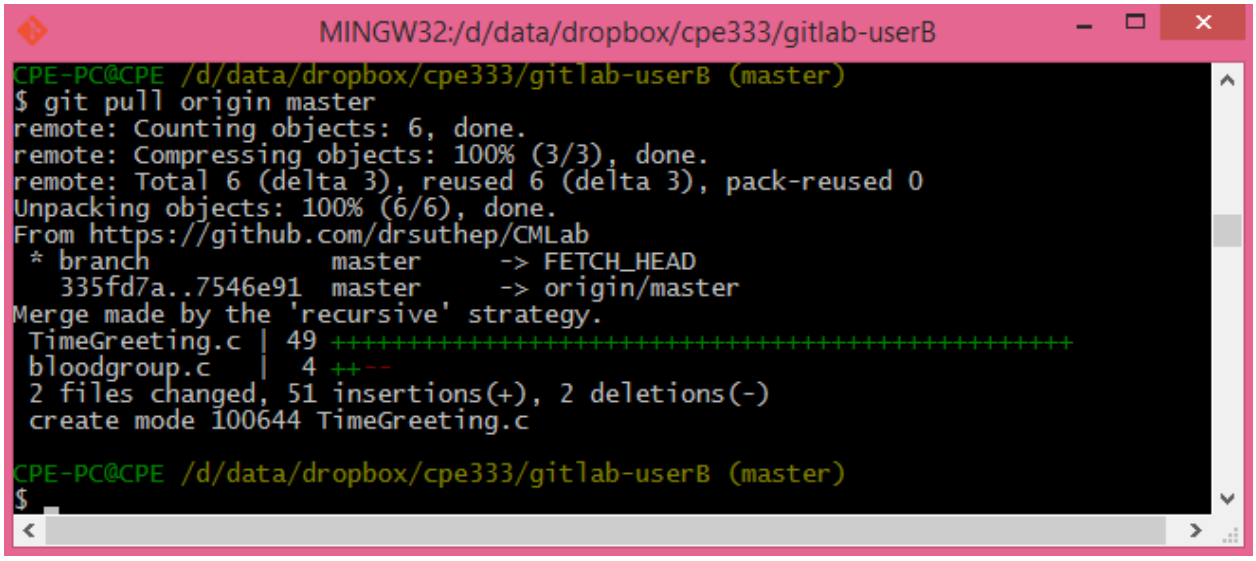
9. User A pushes changes to the GitHub repository. Print hub status. Print User A status.



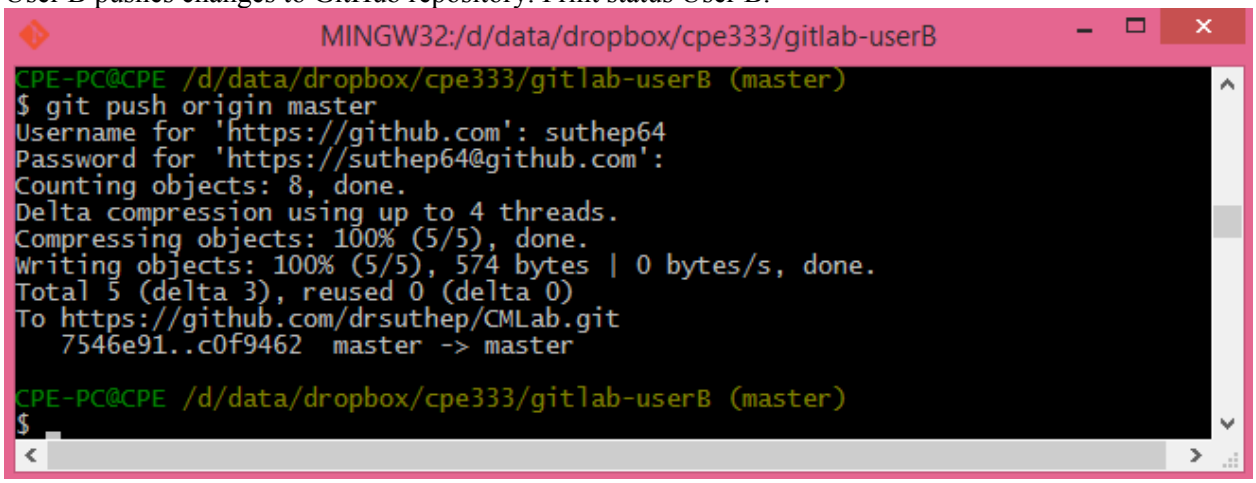
10. User B stages its changes (Program #2 had changed) and commits at once. Print local status User B.



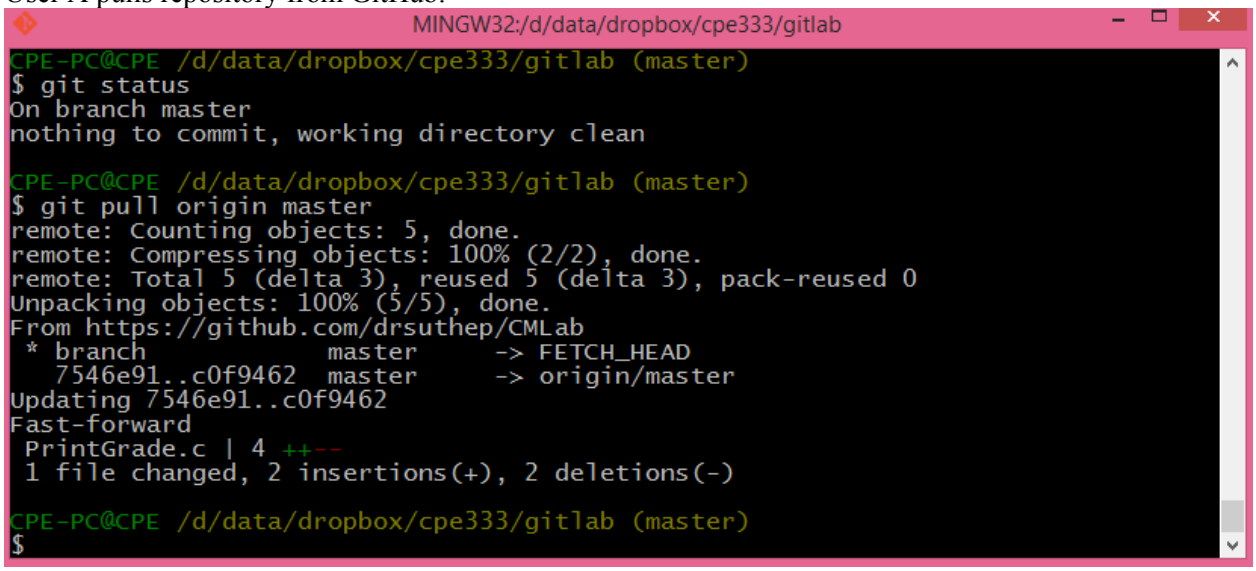
11. User B pulls repository from GitHub.



12. User B pushes changes to GitHub repository. Print status User B.



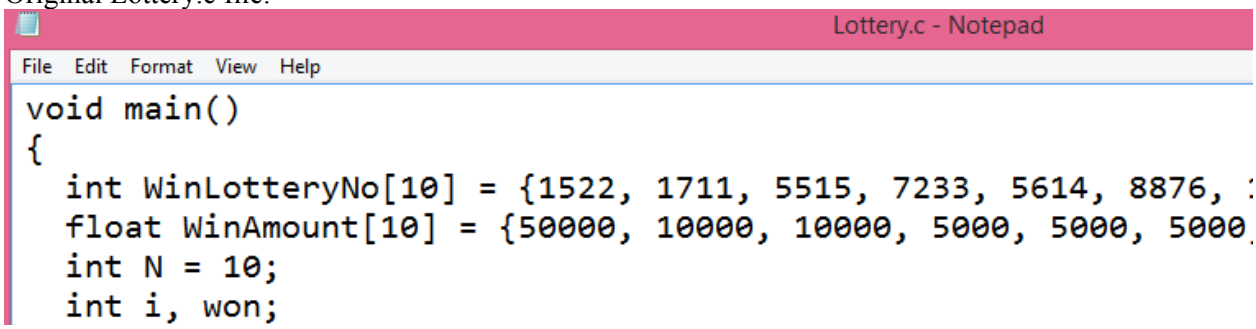
13. User A pulls repository from GitHub.



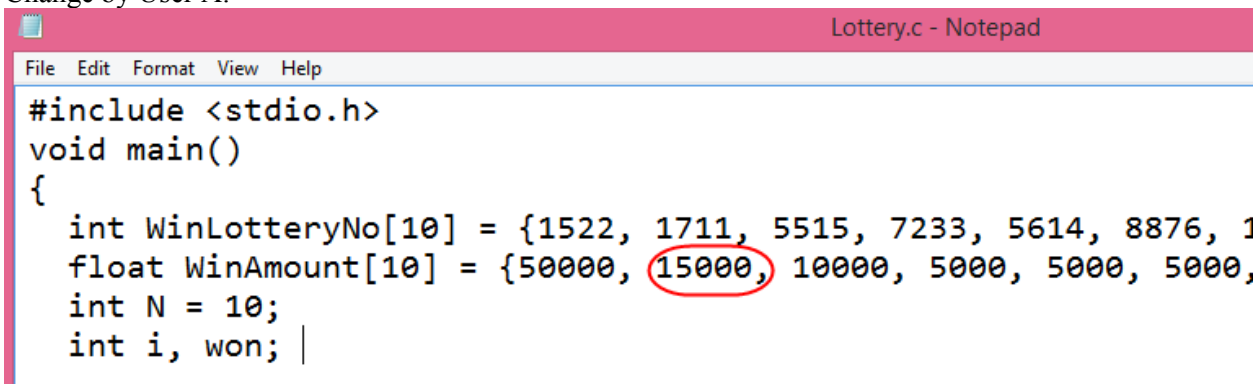
- **All 3 repositories have same copy now.**

14. Then User A changes Program #1 and User B change Program #1, both at a different location. Both add and commit changes in one statement. Print status of User A and User B.

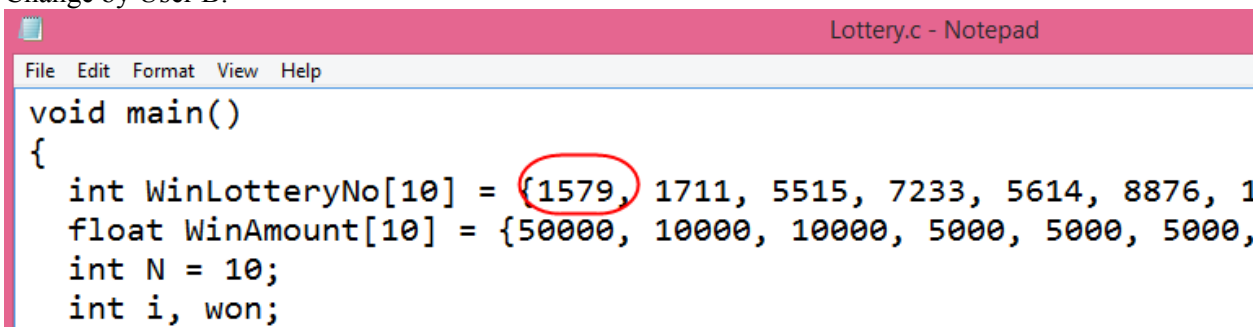
- Original Lottery.c file:



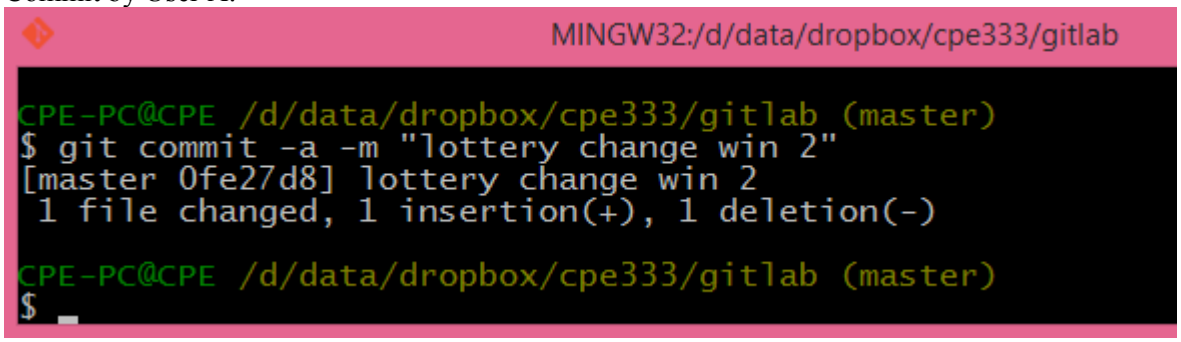
- Change by User A:



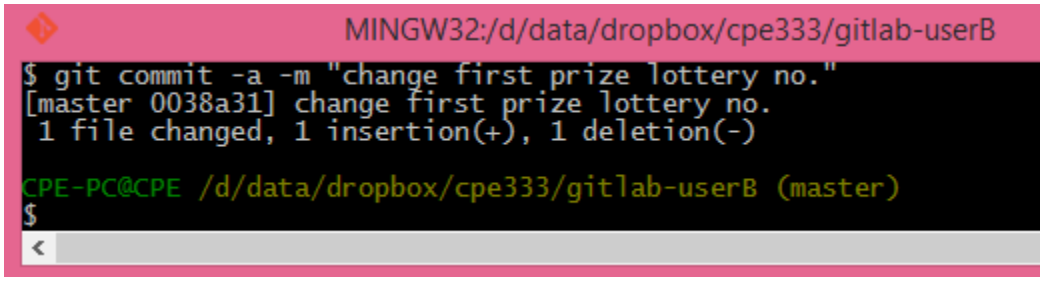
- Change by User B:



- Commit by User A:

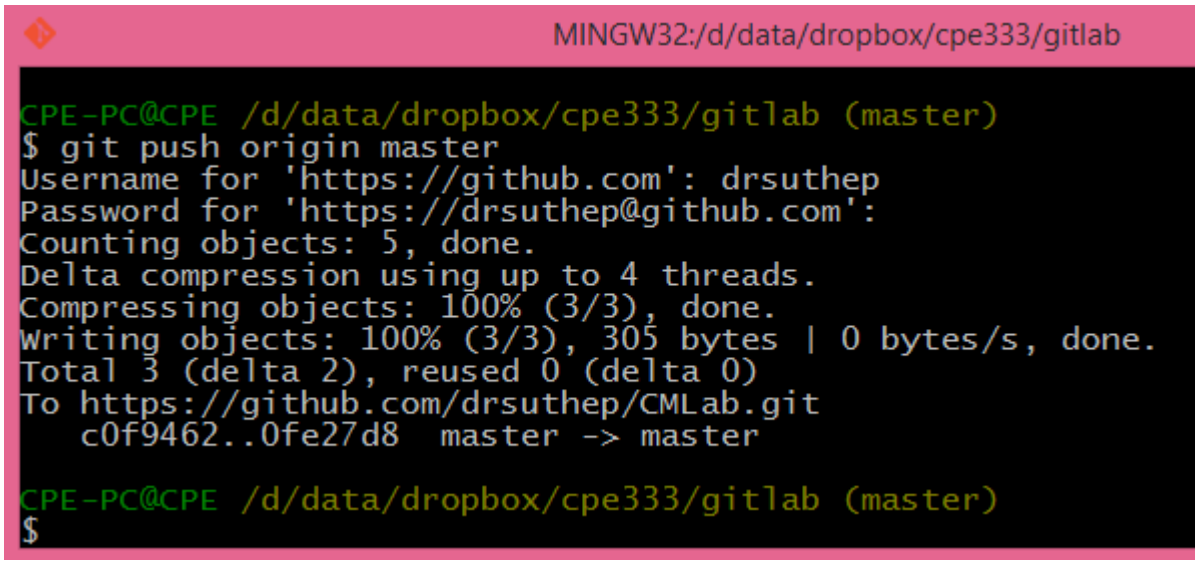


- Commit by User B:

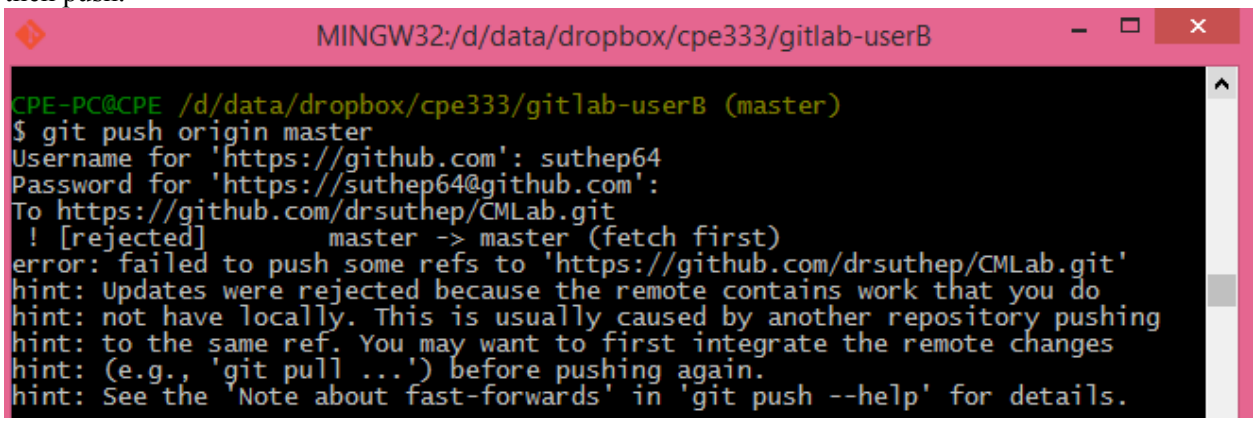


15. They both push changes to GitHub Repository.

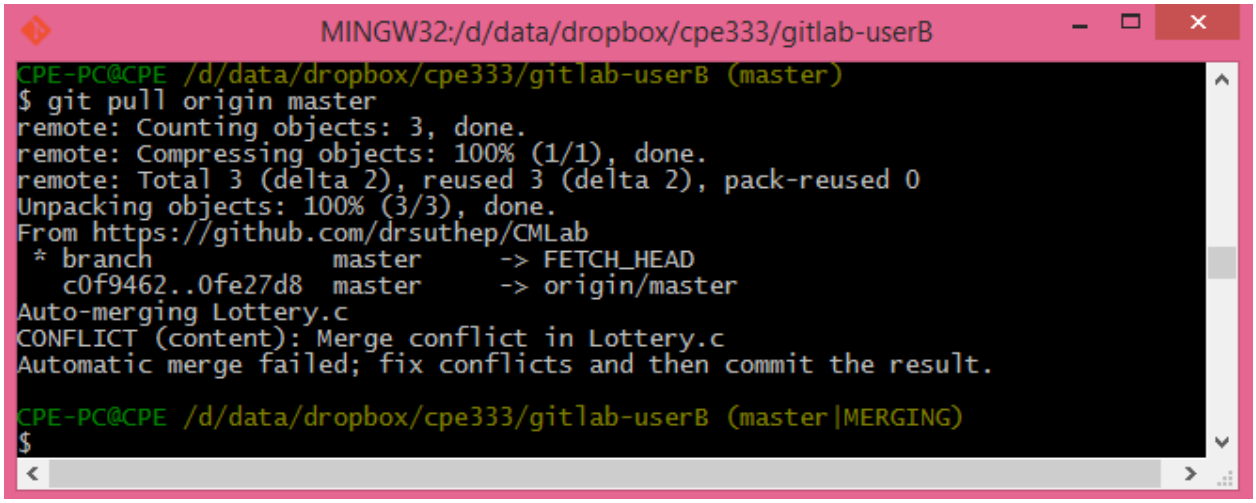
- User A:



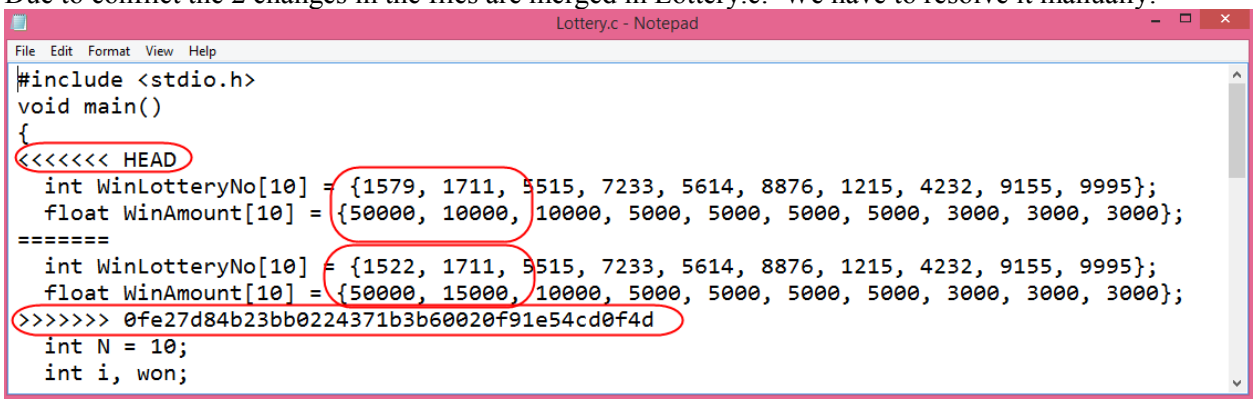
- User B cannot push because some changes were there, so got to pull first. It's always a good idea to pull and then push.



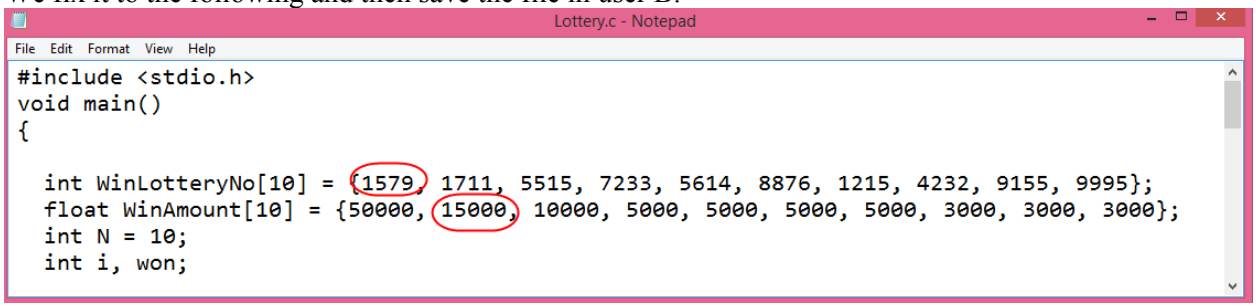
- So Pull first and we realize a merge conflict:



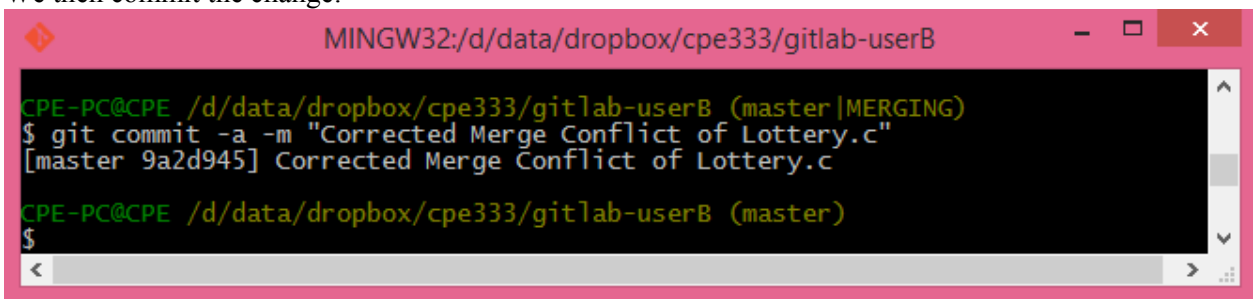
- Due to conflict the 2 changes in the files are merged in Lottery.c. We have to resolve it manually:



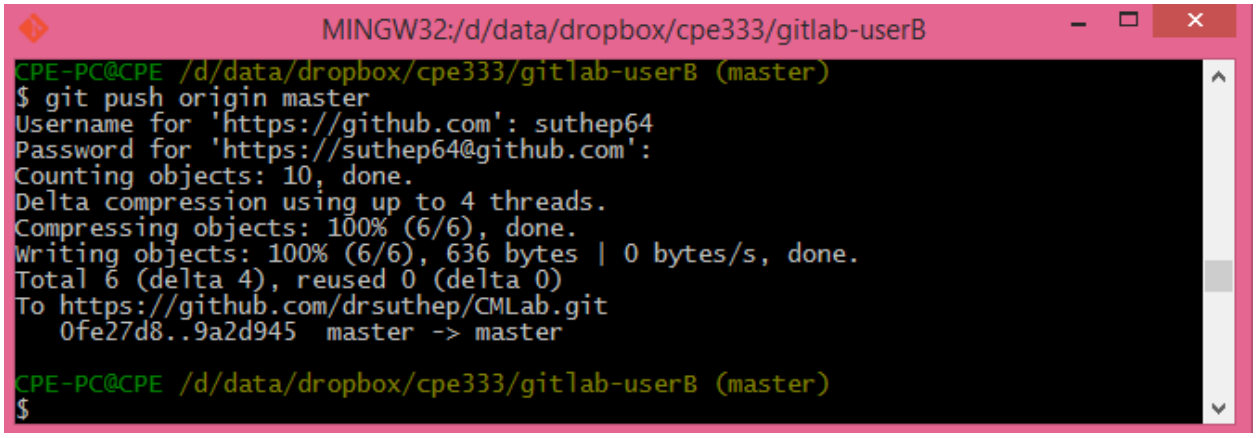
- We fix it to the following and then save the file in user B:



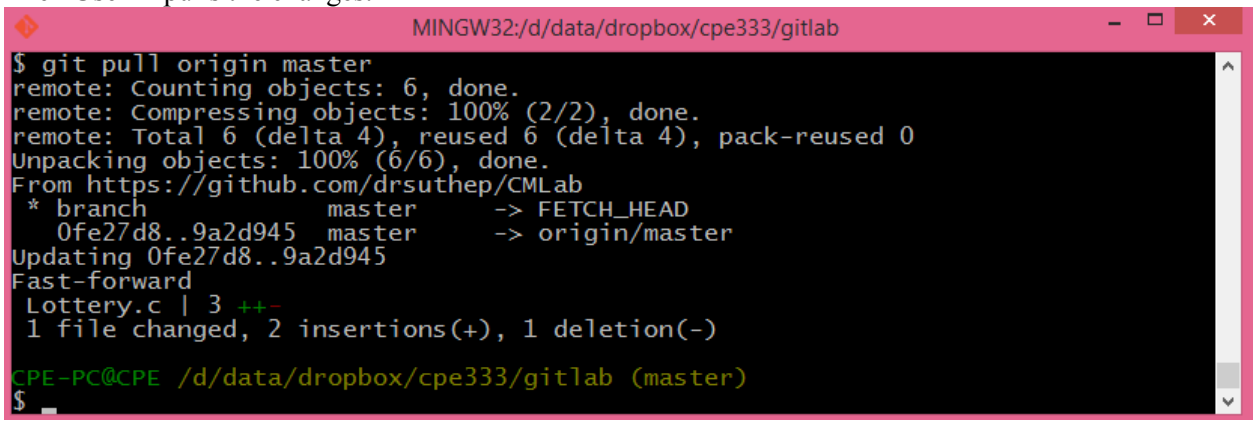
- We then commit the change:



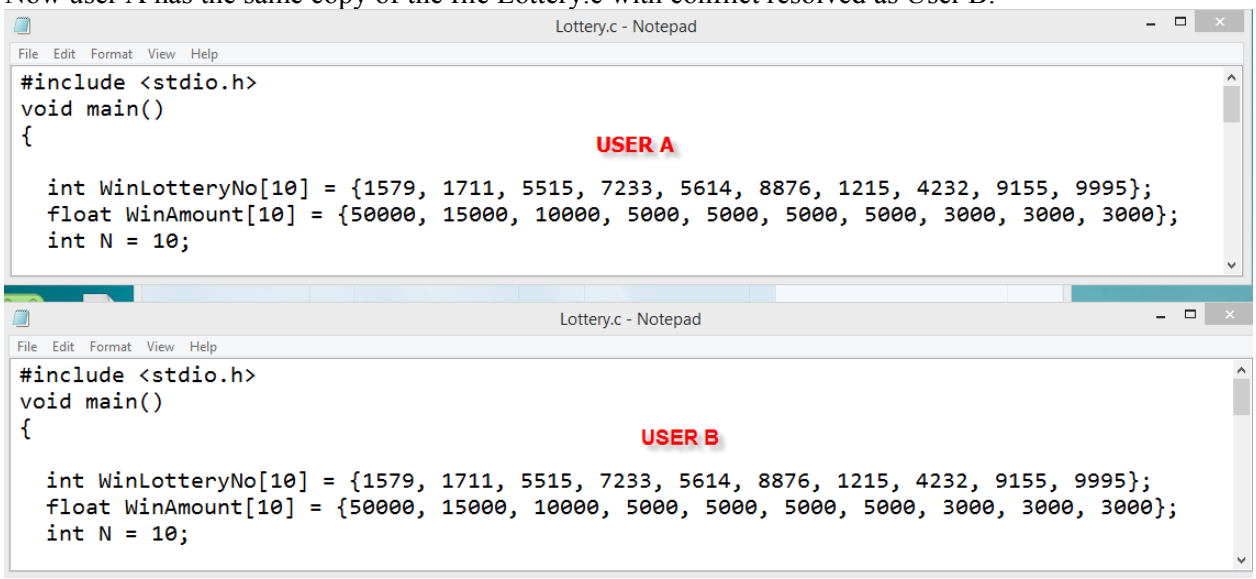
- We then push the changes:



16. Then User A pulls the changes.



- Now user A has the same copy of the file Lottery.c with conflict resolved as User B:



Now Your Turn

Run the following and submit a PDF dump similar to my above workflow dump for the following commands:

1. Create a blank repository in GitHub under User A's account called "GITLAB" with 1 Readme file. Print hub

status. Both users A and B will use this repository. Add User B as collaborator to this repository.

2. In some path create a folder called “CPE333-A” for User A. Make this a git repository and pull data from GitHub. Show status.
3. In some path create a folder called “CPE333-B” for User B. Make this a git repository and pull data from GitHub. Show status.
4. User A. Use file manager to put 2 C programs (Program #1 and Program #2), and 1 MS Word document into CPE333-A.
5. User A. Add all files to git. Then commit. Show Status.
6. User A. Push your local repository to the GitHub repository. (You should always pull before you push). Show screen.
7. User B pulls data from GitHub. Show status.
8. User B change 1 function in Program #2. Print local status User B.
9. User B create a new C program Program #3. Do not put into Git yet. Print local status User B.
10. User B adds Program #3 to Git control (git add). Print local status User B.
11. User B commits changes. Print local status User B.
12. User B pushes changes to the GitHub repository. (You should always pull before you push). Print hub status. Print User B status.
13. User A change something in C Program #1. Print local status User A.
14. User A stages its changes and commits at once. Print local status User A.
15. User A pulls repository from GitHub.
16. User A pushes changes to GitHub repository. Print status User A.
17. User B pulls repository from GitHub.

- All 3 repositories have same copy now.

18. Then User A changes Program #1 at 2 places and User B changes same Program #1 at 2 places (make sure they change same general error or in the next step there may be no merge conflict visible ☺) . Both add and commit changes. Print status of User A and User B.
19. User A pulls from GitHub and then pushes to GitHub. User B then pulls and pushes to GitHub, but there’s conflict. (There will be a conflict if user A and B both change Program #1 at about the same lines). Resolve this merge conflict for User B and commit change. Show each step.
20. Then User A pulls the changes. Show the Program#1 for both users that they are now the same.