Ministerul Educaţiei şi Ştiinţei

al Republicii Moldova

**Universitatea Tehnică a Moldovei**

*Facultatea Calculatoare Informatică şi Microelectronică*

***Tehnologii Informationale***

***Lucrare de laborator nr.1***

***la MIDPS***

A efectuat: st.grupei TI-152

Staver Vasile

A verificat: Cojanu Irina

**Chişinău 2017**

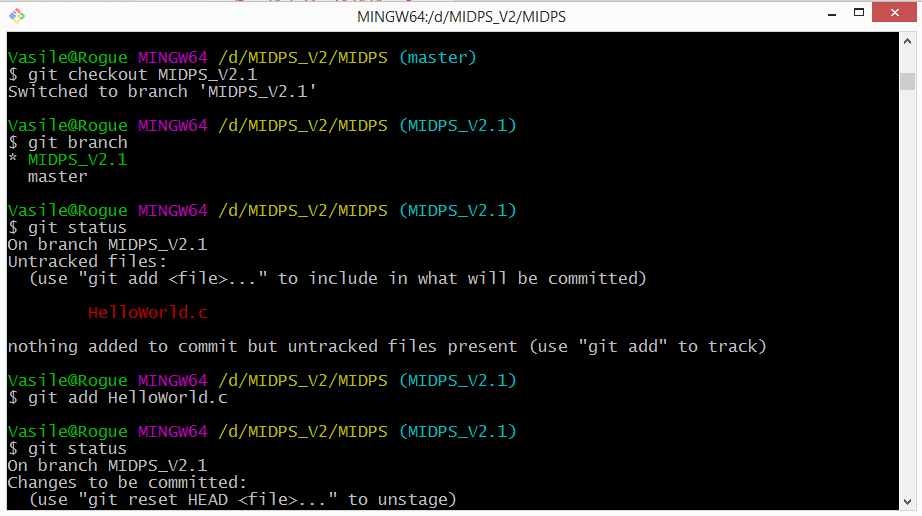
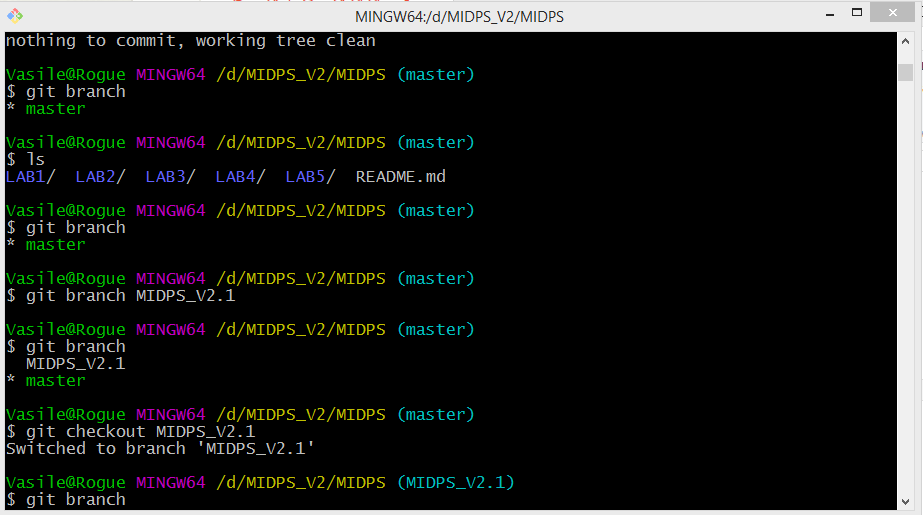
### Obiective:

* Version Control Systems (git || bitbucket || mercurial || svn)

### Laboratory Requirements: Trebuie folosit drept VSC [github](https://github.com/) sau [bitbucket](https://bitbucket.org/)

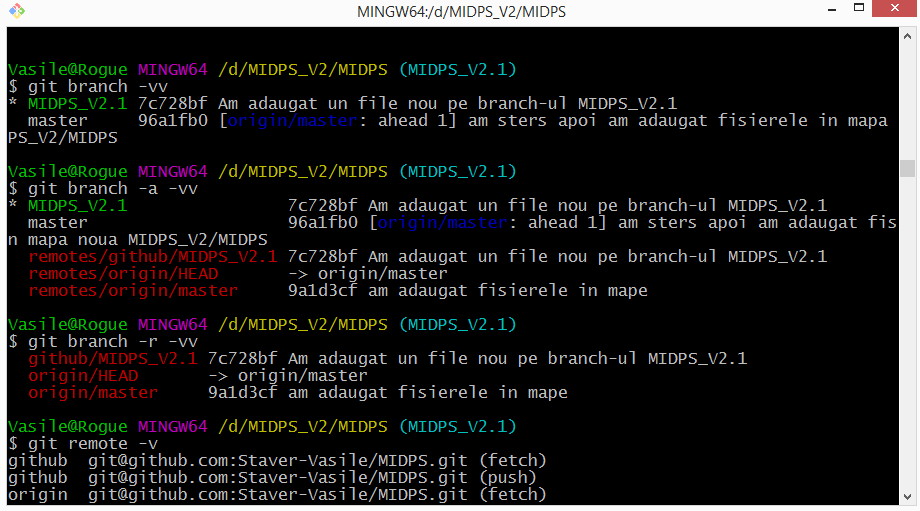
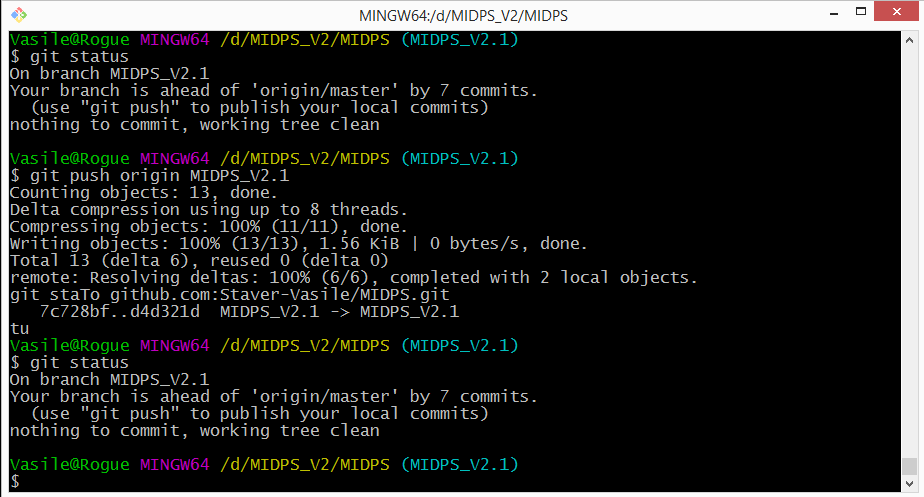
* Basic Level (nota 5 || 6) :
  + initializeaza un nou repositoriu
  + configureaza-ti VCS
  + crearea branch-urilor (creeaza cel putin 2 branches)
  + commit pe ambele branch-uri (cel putin 1 commit per branch)
* Normal Level (nota 7 || 8):
  + seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)
  + reseteaza un branch la commit-ul anterior
  + salvarea temporara a schimbarilor care nu se vor face commit imediat.
  + folosirea fisierului .gitignore
* Advanced Level (nota 9 || 10):
  + merge 2 branches
  + rezolvarea conflictelor a 2 branches
  + [comezile git](https://www.siteground.com/tutorials/git/commands.htm) care trebuie cunoscute
  + ***crearea branch-urilor (creeaza cel putin 2 branches):***

La crarea branch-urilor am utilizat comanda git *branch branch\_name,* am creat doua branch-uri pe care am făcut mai multe commit-uri. Branch-urile le alegeam cu comanda git checkout.Pentru a efectua un commit pe un branch este nevoie să creeam mai întâi un fișier în Directiva noastră de lucru apoi prin intermediul git add să adăugăm acest fișier în zona virtuală Index, cu ajutorul comenzii git commit –m să adăugă fișierul pe repozitoriul nostru locar și prin git pushh origin branch să adăugăm fișieru pe branch-ul de pe remote repozitory(GitHub în cazul nostru)



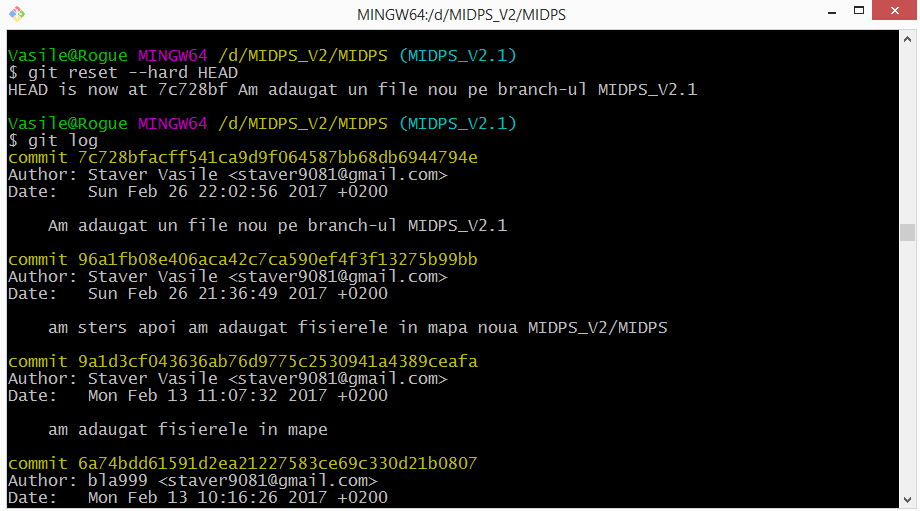
* + ***seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)***

Cu ajutorul comenzii git branch branch\_name --set-upstream-to your\_new\_remote/branch\_name am setat branch-ul MIDPS\_V2 to track a remote origin.



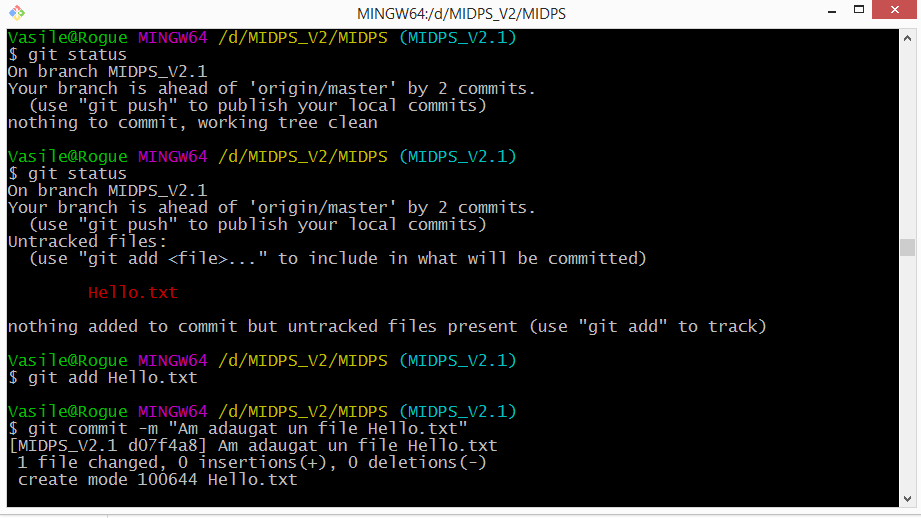
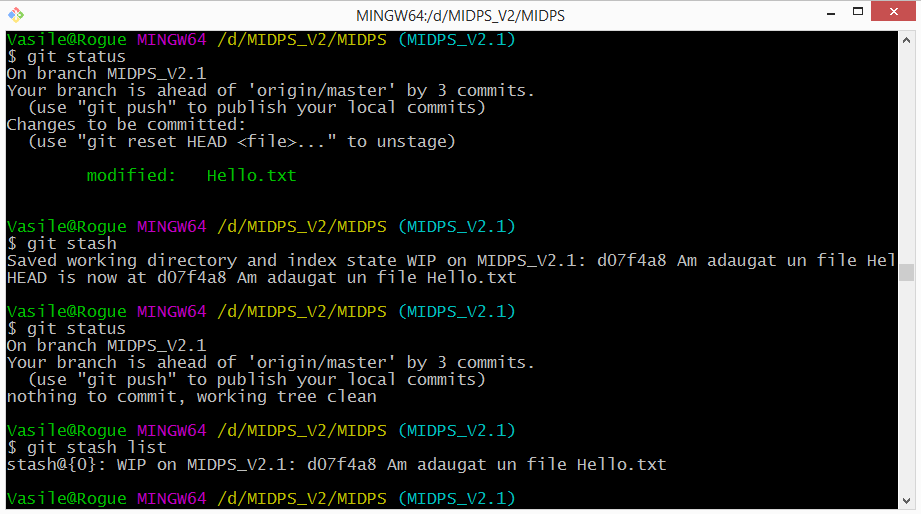
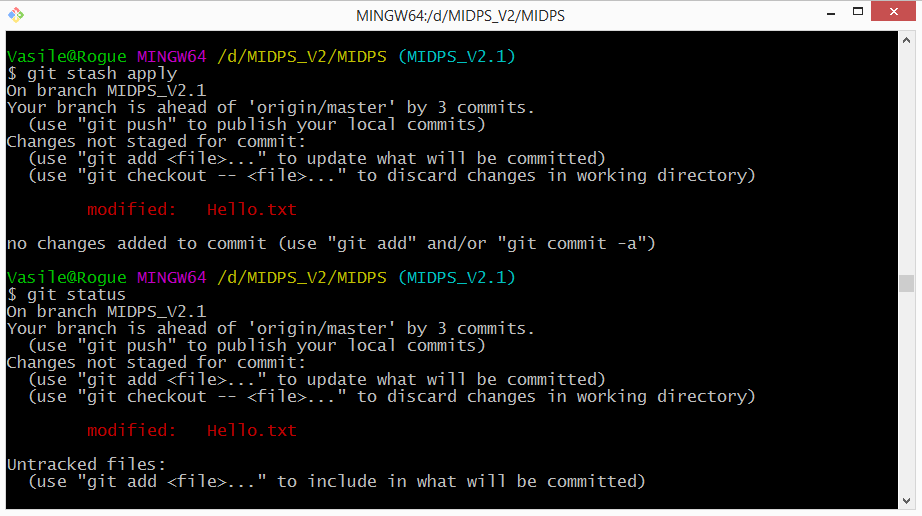
* + ***reseteaza un branch la commit-ul anterior***

Pentru a reseta un branch la commit-ul anterior am utilzat comanda git –gard HEAD



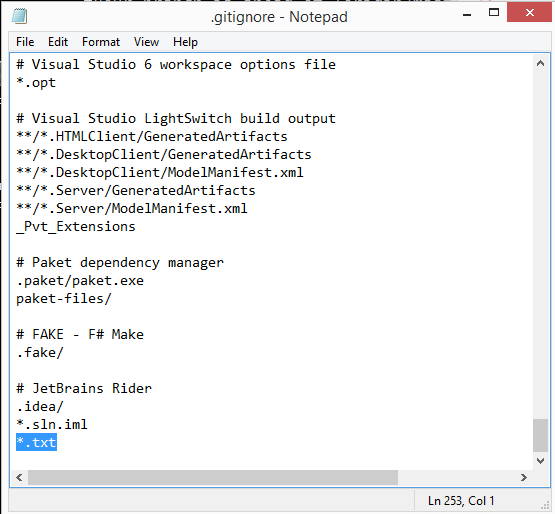
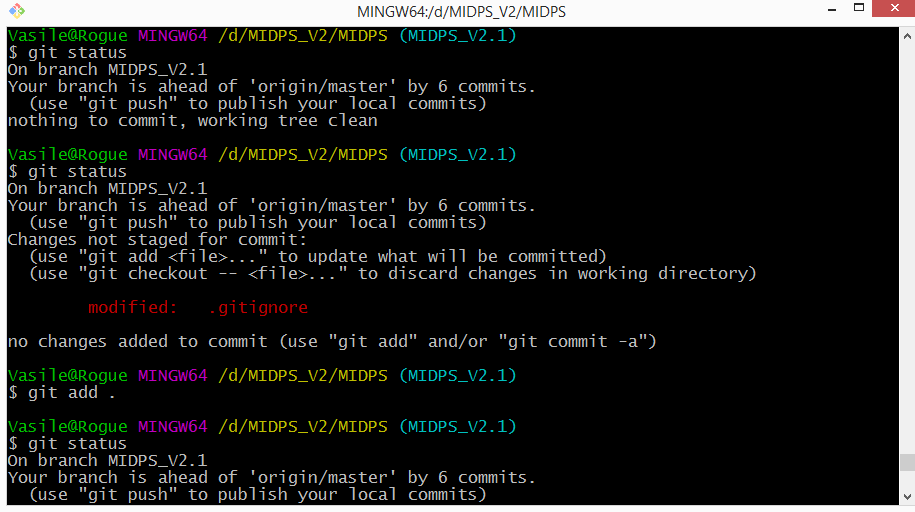
* + ***salvarea temporara a schimbarilor care nu se vor face commit imediat***.

Salvarea temporară a schimbărilor presupune situația în care dacă ai făcut careva modificări într-un file și nu le-ai terminat dar dorești să faci altă sarcină, să treci pe alt branch și ca să nu faci commit la un program neterminat utilizezi comanda *git stash* care salvează totul pe un stack. Cu ajutorul comenzii *git stash list* poți să vizualizezi fișierile care sunt în stack iar pentru a reveni la utilizezi comanda *git stash apply*



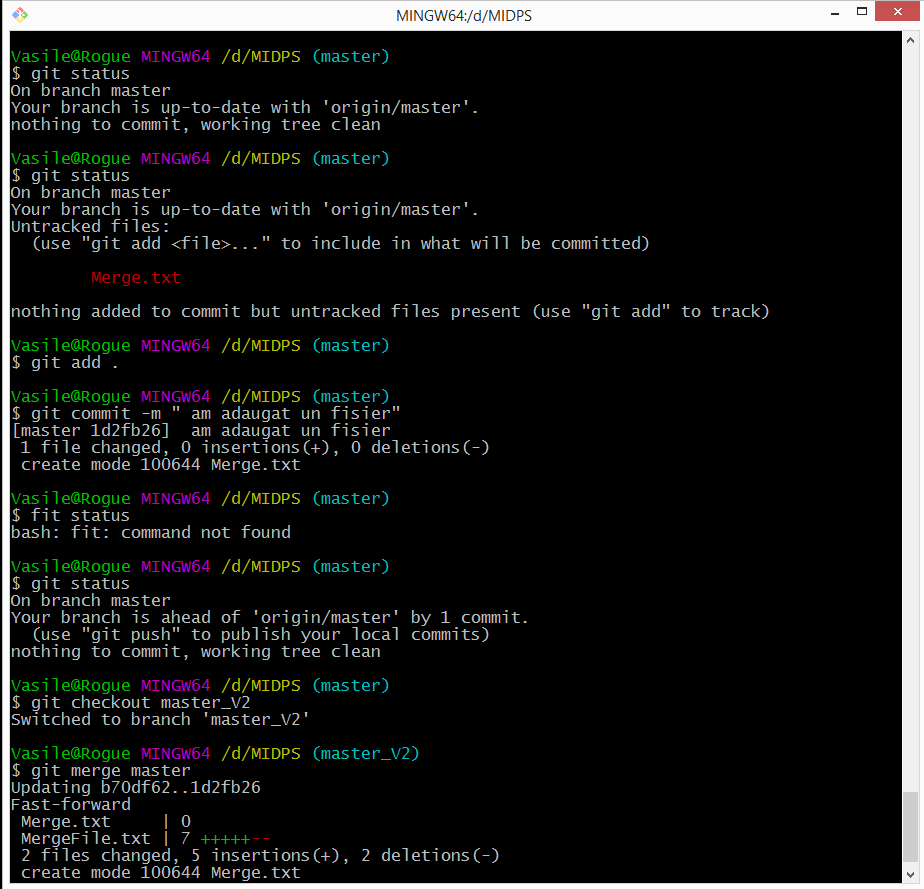
* + ***folosirea fisierului .gitignore***

Pentru toate ,,untracked files” sau fișierele la care nu dorim să facem commit și să apară în repozitoriul nostru este utilizat fișierul .getignore care presupune că toate fișierele sau extensiile din acest fișier sunt ignorate. La realizarea acestei sarcini eu am introdus în fișierul gitignore extensia .txt pentru a demonstra aceasta proprietate.



* + ***merge 2 branches***

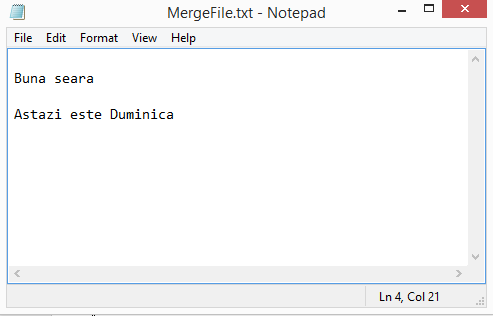
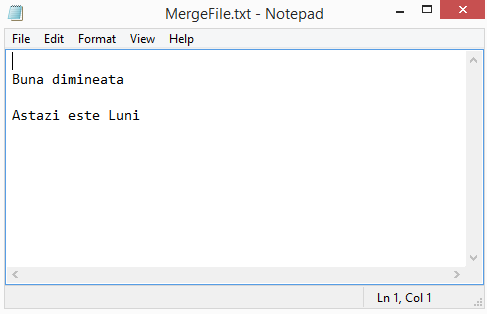
Merge-ul este una din cele mai importante funcții în Git care permite să copierea tuturor fișierelor(sau conținutul fișierelor cu același nume) dintr-un branch în altul.Acest lucru permite lucrul mai multor echipe asupra aceluiași program pe branch-uri diferite să fie unit cu ajutor-ul merg-ului într-un program final.La realizarea aceste-i sarcin eu doar am făcut merge între doua branch-uri identice cu excepția unui singur fișier ceea ce a permis ca fișierul dat să fie copiat în branch-ul în care acesta lipsea.

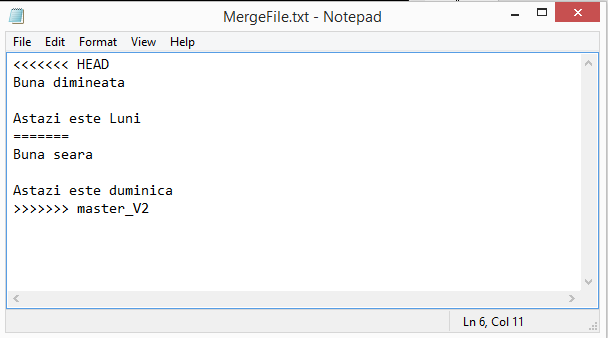


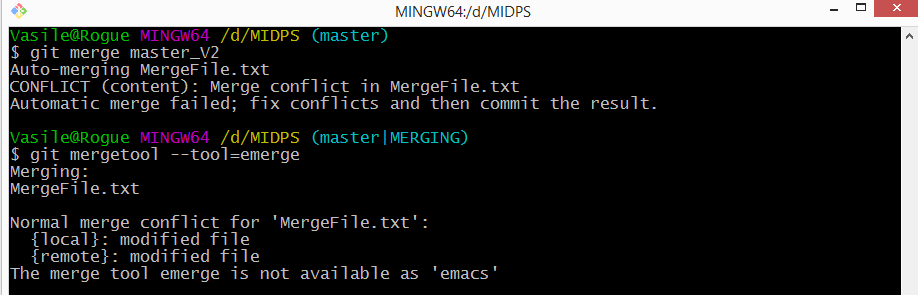
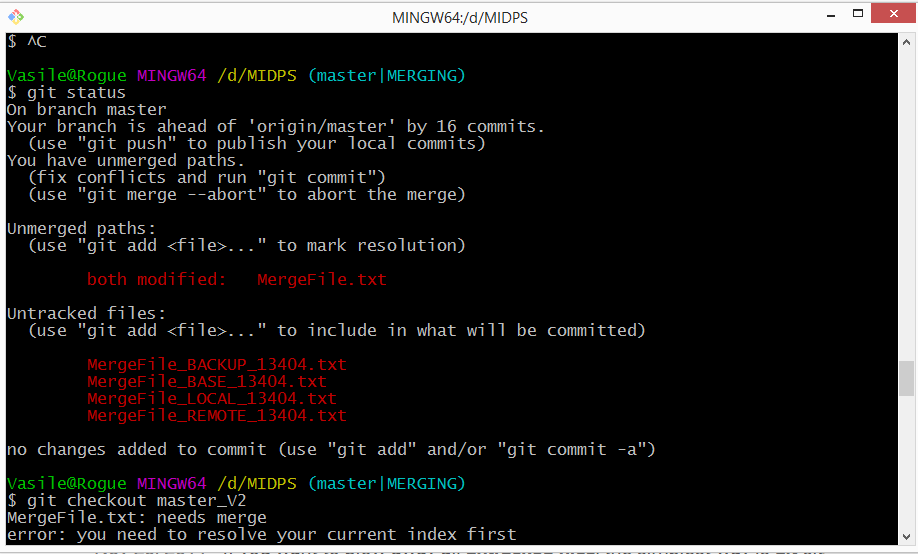
* + ***rezolvarea conflictelor a 2 branches***

Conflictele între doua branch-uri apar atunci când același fișier pe aceleași rânduri este modificat în ambele branch-uri sau un același fișier într-un branch este șters iar în al 2-lea modifcat și când are loc merge-ul git-ul nu știe care variantă trebuie să fie aleasă.Eu pentru a crea conflicte am modificat același fișier în ambele branch-uri. Sunt mai multe metode de a rezolva conflictele, eu am utilizat metoda manuală.

***Fișierele din cele 2 branch-uri:***

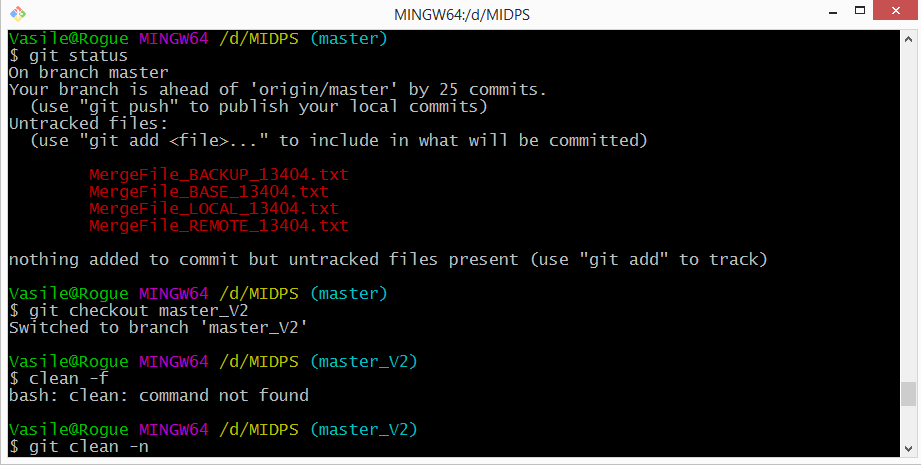
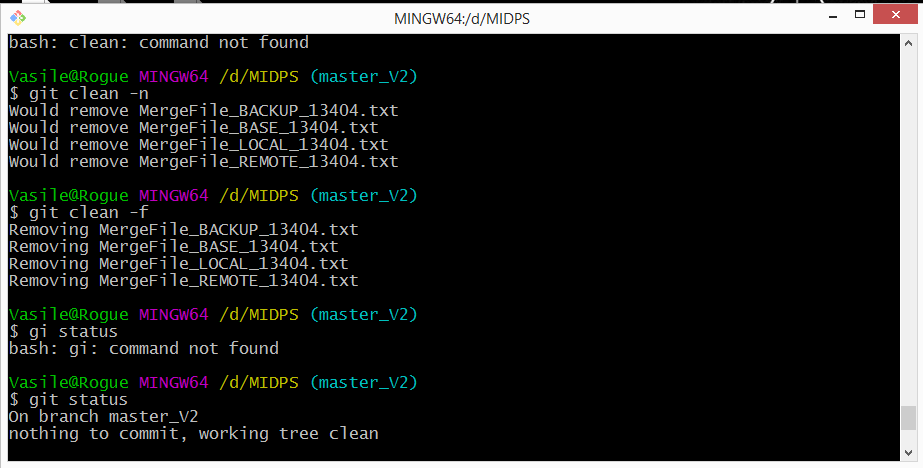
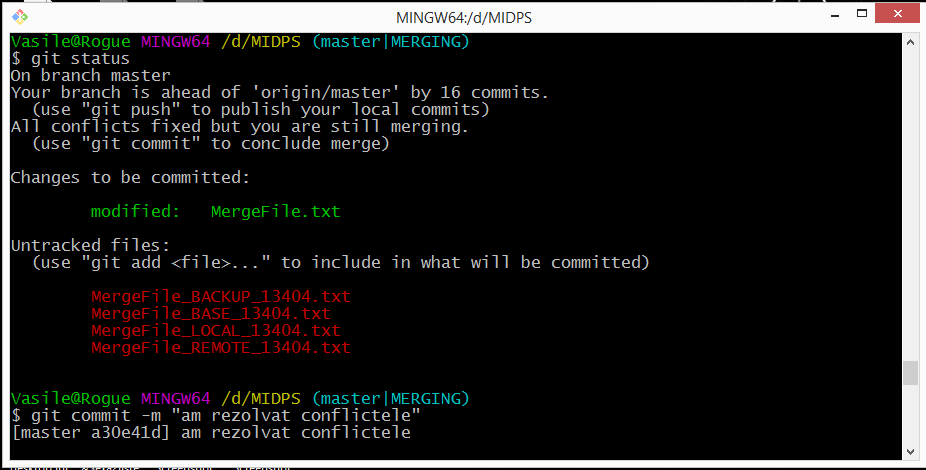


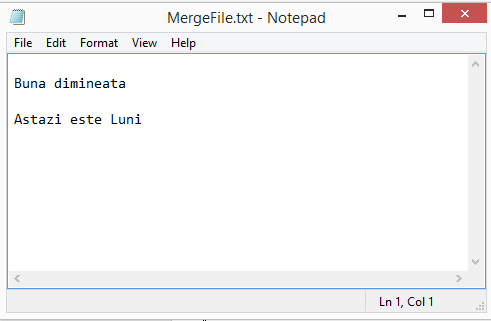
***După efectuarea merge-ului:***



În cazul dat pentru soluționarea conflictului trebuie să alegem una din variantele despărțite de ,,=======’’ și să ștergem rândurile care au apărut în urma merge-ului.

***Astfel obținem fișierul***:





***Concluzie:***

La efectuarea acestei lucrări de laborator eu am făcut cunoștință cu sistemul revision control ,,Git” și cu GitHub-ul este un [serviciu de găzduire web](https://ro.wikipedia.org/wiki/G%C4%83zduire_web) pentru proiecte de dezvoltare a software-ului care utilizează sistemul de control al versiunilor [Git](https://ro.wikipedia.org/wiki/Git_(software)). Am înțeles că git-ul reprezintă un instrument extrem de util pentru scrierea și gestionarea programelor largi și realizării versiunilor. La realizarea sarcinilor propuse am studiat proprietățile de creare a repozitoriilor locale cât și ,,remote”.Am analizat comenzile de bază a git-ului așa ca *add și commit*. Am învățat așa termeni ca ,,Working directory” care reprezintă mapa în care noi lucrăm, ,,Index” –ul care reprezintă o zonă virtuală în care noi încărcăm toate fișierile și modificările respective ca mai apoi prin ,, commit’’ să încărcam aceste fișiere în repositoriul nostru local. Deasemenea exista și ,,Remote repository” care în cazul nostru reprezintă Github-ul astfel prin intermediul unor comenzi ca *push, fetch, pull* putem comunica între aceste repozitorii.Deasemenea am studiat branch-urile care permit unor persoane diferite să dezvolte același soft independent unii de alții și ținând cont de toate modificările anterioare și să întrunească într-un final prin așa comandă ca *merge* toate proprietățile unui program efectuat anterior de grupuri de oameni diferiți. Deasemenea am văzut și am implementat conflictele care pot aparea între 2 branch-uri și modalitâțile de soluționare a acestora.

Bibliografie:

- google.com

- youtube.com

- https://git-scm.com/book/en/v2/