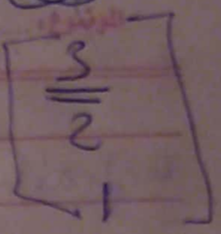
git diff onhu on[3] _a (ISr< b.y)

a b

Yehia Qarele

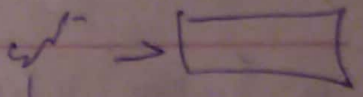


git diff an[3] an[2]

stg ol +

++

1] Push.sh → Python Script : library in python to get ".git.logs"



2] Call branch function (view) to get last global Commit hash from array of hashes →

Master[Master.length _ 1].hash

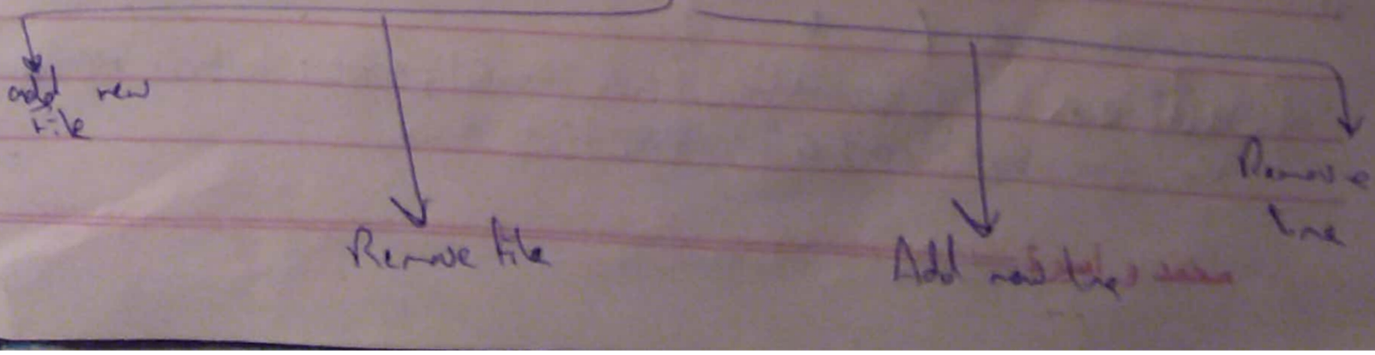
3] If the two hashes are equal ⇒ Khlas

4] If not equal ⇒ While (random not equal)

↳ Hankaren el .git diff bad el global be aser wahed fel array

↳ law mesh zayo benaamel ne execute Command lehad manewsalo we ben push fe array el Output elly ~~bygelha3~~ bygelha3 we fe aser Kol output bnekteb "end_Commit" 3alshan Ne split beha

→ Hanemshy 3ala el array we haygelna 3andena 4 Cases



Solidity

```
Contract createRepo{

    array   repos[]
    map

→ function betrady constructor elnew repo
                                            address
    parameters(repoName, description), array of collaborators)
})
_____

Contract Repo{
  name                    , date
  desription              , branches[]
  owner                   , map []
  collaborations[]        , CollaboratorMap (address → bool)

  → Constructor(name, desription, Collaborators ){
        owner = msg.sender
        dateCreated = now
        parameters =

create ~~File~~ ~~Rom~~ newBranch,, master = [map w array fatyeen]
      • map ["master] = masterBranch
    branches. push(master);
}
}
function add Collaborator (address collaborator) only Owner
      Collab. push (                )
   { emit event CollabAdded (address coll, ~~Add~~this.address, repoName)
   {Collaborator Map (collaborator) = true;
```

التاريخ :..................

الموضوع :..................

```
function removeCollab( addres, Coll) onlyOwner
        CollabMap( Coll ) = false
        emit  CollaboratorRemeved ( address , RepoAddress , Name )

  function  makeBranch ( branchName ) onlyPermitted {
        Branch  newBranch = new Branch(branchName, branches[0].map,
                                                branches[0]. array);

         branches.push(newBranch);
          map.push (newBranch);
         emit branchCreated ( repoName, branchName);

  }
```

التاريخ :

الموضوع :

Branch

```
Contract Branch {

    branchName,
    CommitMap [CommitHash => Contract]
    CommitArray []


  Constructor ( branchName, CommitMap, CommitArray )


  ~~[illegible] / makeCommit~~ (

                            hash
   function   parseCommit ( , date, branch, msg, change) onlyPermitted

        Commit newCommit = new Commit (msg.sender,
                        rets el fnC  parameters);

        ~~CommitMap~~   CommitMap.push (newCommit);
                       CommitArray.push (newCommit);

      emit CreatedCommit ( msg.sender , msg, hash);

 3
  ~~function~~ funtion  makeRoot (
       if ( ( commArr.length _1 == 0)
      -> c  parseCommit (____ ___ )
         push
Contract Commit {



function initCommit(            , EnterProject)


        chars= EnterProject;   }



                                  view
        getLastCommitHash(){

        return Commits[commits.length-1].hash]; }
```