```
git commands
#set a global configuration in local
        git config --global user.name <name>
        git config --global user.email <email>

#for check the repositry
        git checkout https://github.com/Adi8090/project.git

#for cloning that repo in local
        git clone https://github.com/Adi8090/project.git

#modify any of the file in local
        vi <filename>
                for saving file :wq!

#adding changes oin gilt local terminal
        git add .

#check the status of changes in local
        git status

#create a commit for change in file which will pushed
        git commit -m "push it"

#push all changes to the repo(git hub)
        git push

#check all the commits have done
        git log
 or     git log --oneline
                //where all commits are listed with the commit id
        git log <limit>
                // show the limited no. of commit  which you prefer in limit

#revert your commit with the help of commit id
        git revert <commit id>
                //if revert fail or some how its not working it can be abort or
skip
                git revert --skip
                git revert --abort

#if you have multiuser repositry where changes done by multiple
 user  always pull the repo before doing the push
        git pull
        // git pull is used for to clone all latest updated file in local

#how to knw the global config and edit it
        git config --global --edit

--------------------------------------------------------------------------------
-------------------------------
Create conflics and resolve conflicts

#create a direcotry
  mkdir git-merge-test

#change direcotry ehich created
  cd git-merge-test/

#initiate git repo in directory
  git init

#make a file in this direcotry
  echo "this is some content to mess with" > merge.txt
```

```
#add changes
        git add merge.txt
        git add .
#check status
        git status
#commit it
        git commit -am "we are commiting the initial content"

#create new branch
        git checkout -b new_branch_to_merge_later
#create new file in new branch with same name
        echo "totally diffrent content to merge later" > merge.txt
#add changes
        git add merge.txt
        git add .
#check status
        git status
#commit it
        git commit -am "editied the content of merge.text to cause a conflict"

#go back to main branch
        git checkout main
#edit again this file with some add-ons
  echo "conten to append" >> merge.txt
#add changes
        git add .
#commit it
        git commit -am "append content to merge.txt"
#try to merge this file merge.txt
git merge new_branch_to_merge_later
        //A conflic apperas
        you can check the status----- git status

#now conflict is created in two diffrent branches with the same file name with
diffrent data

#resolve the conflict

#check whats exactly happend here with help of cat
        cat merge.txt
        //      <<<<<<< HEAD
                this is some content to mess with
                conten to append
                =======
                totally diffrent content to merge later
                >>>>>>> new_branch_to_merge_later

#now open your fav editior and remove head tail comments and divident line
        vi merge.txt
        //      this is some content to mess with
                conten to append
                totally diffrent content to merge later

#add changes
        git add .

#commit it
        git commit -m "merged and resolved the conflict in merge.txt"

conflict is resolved........
```