*OSOC INDUCTIONS TASK*

**Q1. Command for searching pattern line by line in any document with an example.**

Ans:- ‘**grep**’ is the command for searching a pattern line by line in a document.

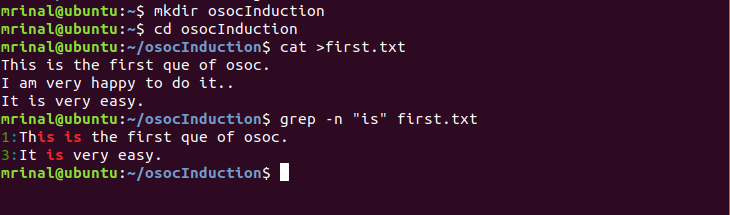
**Syntax:** grep [options] pattern document

Eg:- **grep -n “is” first.txt**

Here **‘–n’** is the option for checking pattern line by line

**“is”** is the pattern specifying line must have “is”

**‘first.txt’** is the file name



**Q2. What all permissions are there in Linux? State all permissions and different way of changing permissions with example**.

Ans:- There are three types of permissions

**Read (r) :-** to enable users to read the file

**Write(w)** :- to enable the users to write or make changes to file

**Execute(x)** :- to enable the users to execute the file

There are three types of users

**User**:- owner of the file

**Group**:- group of the user

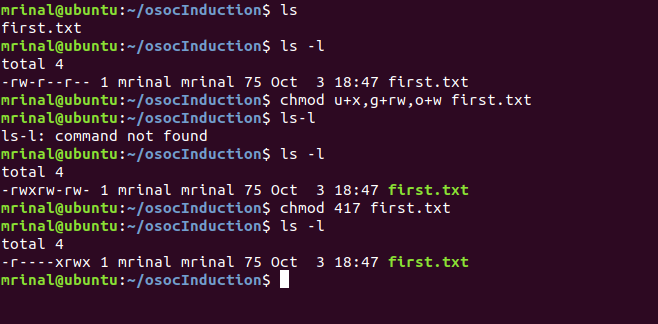
**Others**:- all the others

Ways to change the permissions

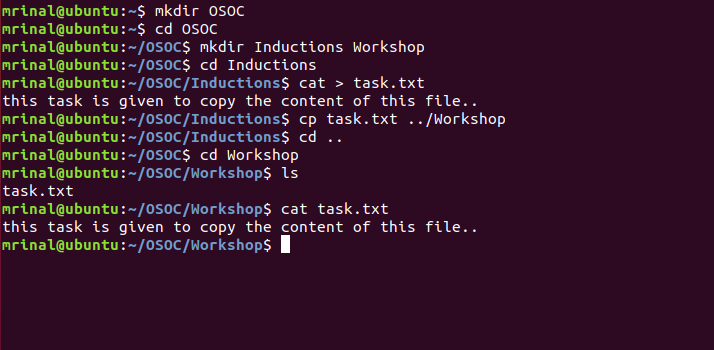
We can change the permission using ‘chmod’ command, ‘+ ’ is used for adding the permission, ’-’ is used for removing the permission and ‘=’ is used for assigning the permission sequence.

1.using 9 character long sequence of r,w,x.

2.using octal no.



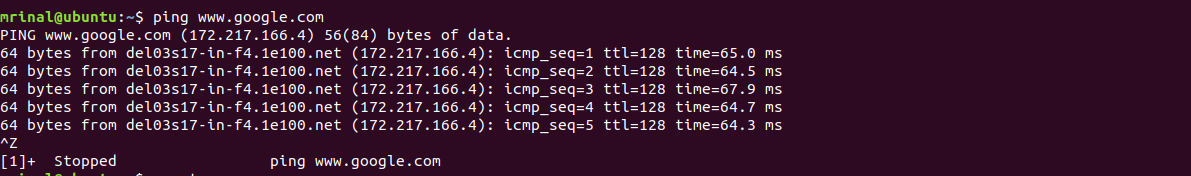
**Q3. Create a folder OSOC and inside that another folder Inductions and now create a file task.txt inside Inductions with some content now copy this file into another folder Workshop in OSOC.**

Ans:- 

**Q4. I want to check whether my system is connected to any network or not, please suggest appropriate command for this.**

Ans:- ping command can be used to check whether system is connected or not

Syntax: ping website\_name



**Q5. I want to change password of another user how will I do that?**

Ans:- passwd username or sudo passwd username.

Eg:- passwd abc



**Q6. Using shell scripting write program for sort an Array. Input should be taken from user**.

Ans:- #!/bin/bash

echo "Enter the no of elements"

read n

echo "Enter the elements"

for((i=0;i<n;i++))

do

read a[$i]

done

for((i=0;i<n-1;i++))

do

for((j=i+1;j<n;j++))

do

if((a[i] > a[j]))

then

temp=${a[i]}

a[$i]=${a[j]}

a[$j]=$temp

fi

done

done

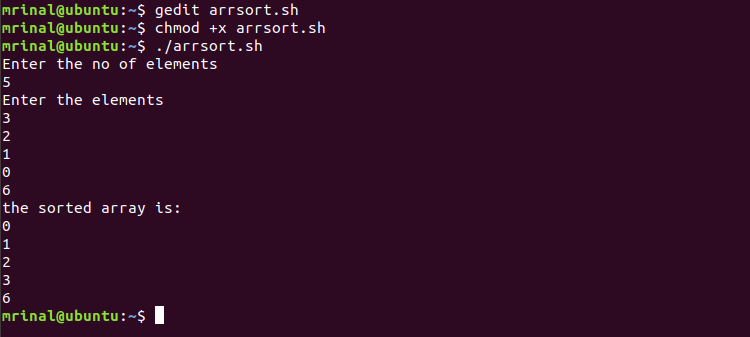
echo "the sorted array is:"

for((i=0;i<n;i++))

do

echo ${a[i]}

done



**Q7. Using Shell scripting write a program to reverse a string. Input should be taken from user.**

Ans:- #!/bin/bash

echo "Enter the string you want to reverse"

read str

len=`echo $str | wc -c`

len=`expr $len - 1`

rev=""

while test $len -gt 0

do

rev1=`echo $str | cut -c$len`

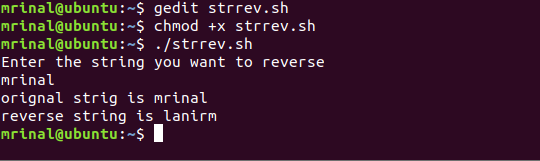
rev=$rev$rev1

len=`expr $len - 1`

done

echo orignal strig is $str

echo reverse string is $rev



**Q8. What is the concept of Branching? How it is useful? Explain with working example.**

Ans:- Branches in Git are nothing but pointers to a specific commit. Git generally prefers to keep its branches as lightweight as possible.

There are basically two types of branches viz. **local branches** and **remote tracking branches**.

A local branch is just another path of your working tree. On the other hand, remote tracking branches have special purposes. Some of them are:

* They link your work from the local repository to the work on central repository.
* They automatically detect which remote branches to get changes from, when you use **git pull**.



**Q9. What is Push, Pull and Commit? Explain with example**.

Ans:- **Push** :- The git push command is used to transfer or push the commit, which is made on a local branch in your computer to a remote repository like GitHub.

git push 'remote\_name' 'branch\_name'

**Pull:-**  If you make a change in a repository, GIT PULL can allow others to view the changes. It is used to acknowledge the change that you've made to the repository that you're working on. Or also called a target repository.

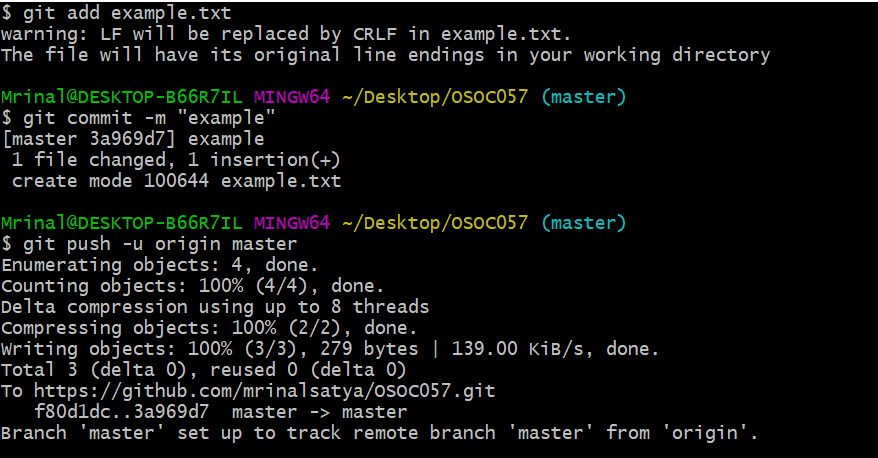
The simple command to PULL from a branch is:

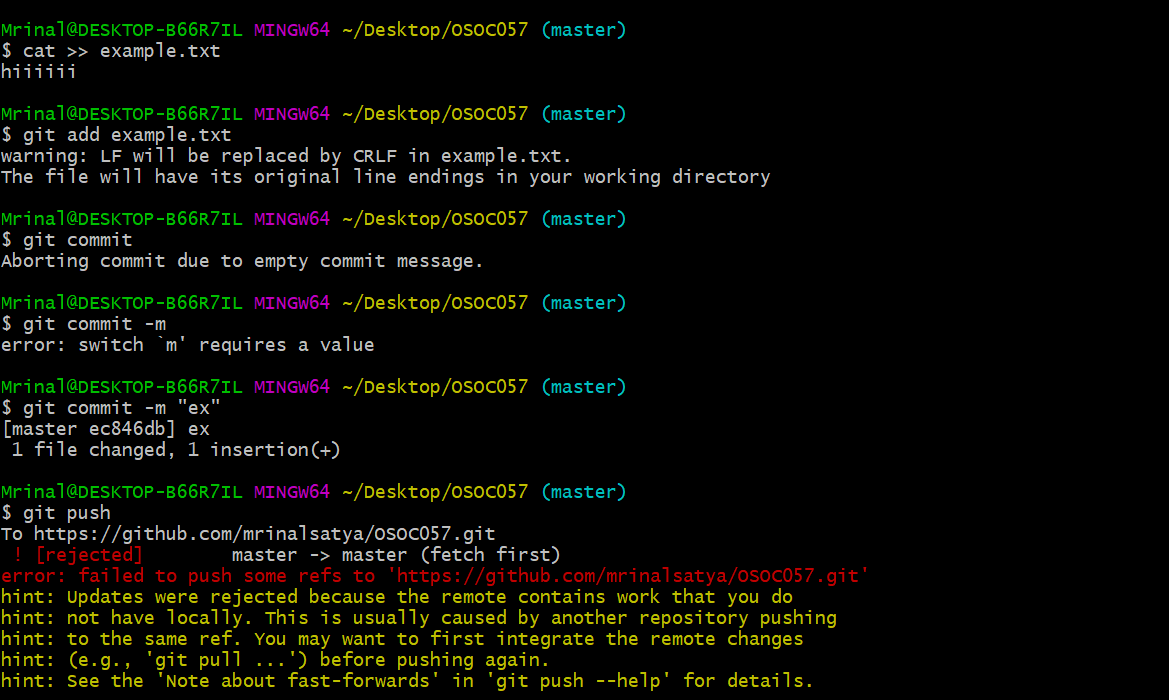
git pull 'remote\_name' 'branch\_name'

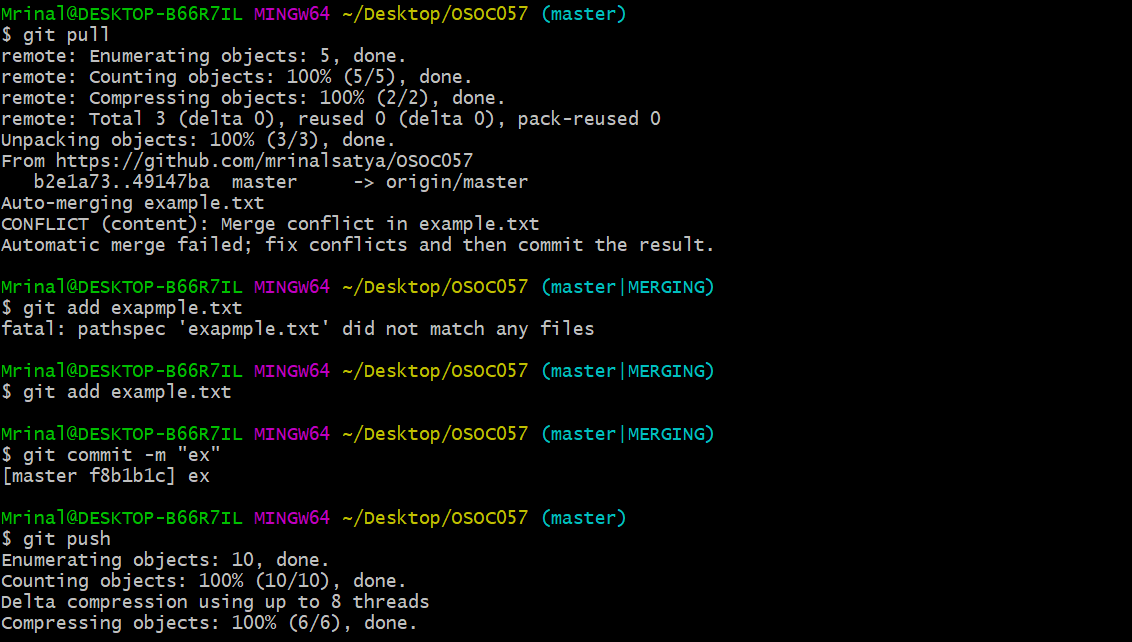
**Commit:-** The "commit" command is used to save your changes to the local repository.

Syntax:git commit –m file name

-m: **Sets the commit's message.**







**Q10. Explain how we can restore a file after a commit with an Example.**

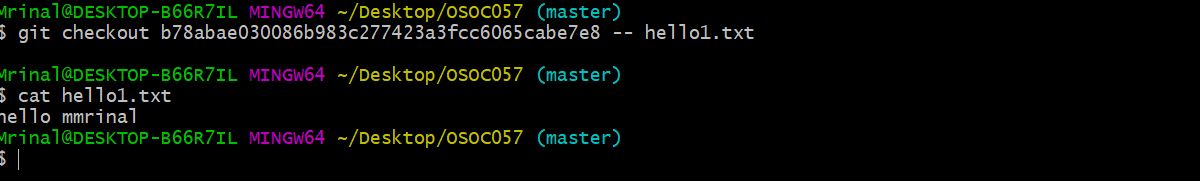
Ans:-

**Revert:** A revert is an operation that takes a specified commit and creates a new commit which inverses the specified commit. git revert can only be run at a commit level scope and has no file level functionality.

**Checkout:** 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.

Syntax: git checkout <commit\_ID> path/to/the/file.txt

git revert <commit\_ID>

****

**Q11. How to preview the changes you have made before applying merging command? (write optimized command).**

Ans:-

**Q12. How to apply any commits of current branch ahead of specified one? (write command with screenshot of command line).**

Ans:-

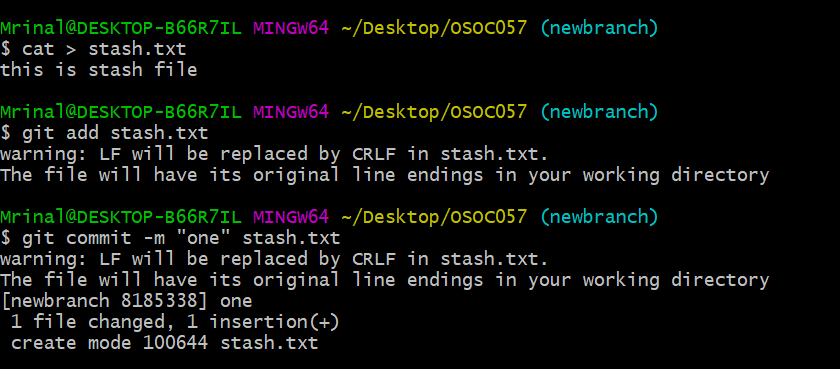
**Q13. what is stash stack? How to write working from top of stash stack? (write answer and show screenshot of command on command line).**

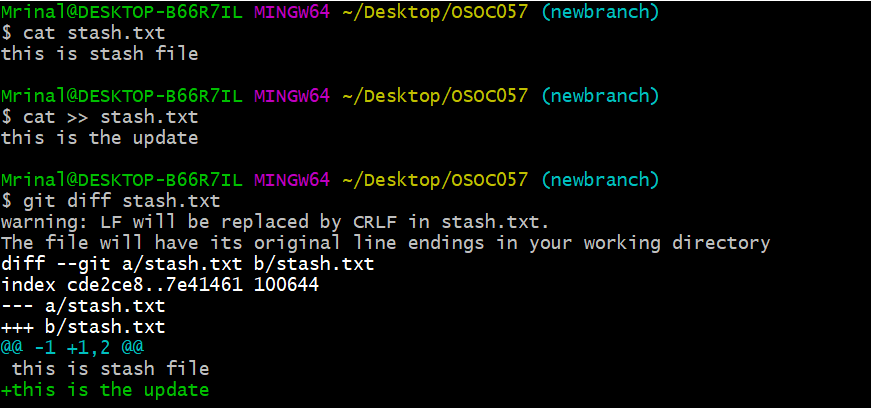
Ans:- **Stashing :** git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on. Stashing is handy if you need to quickly switch context and work on something else, but you're mid-way through a code change and aren't quite ready to commit.

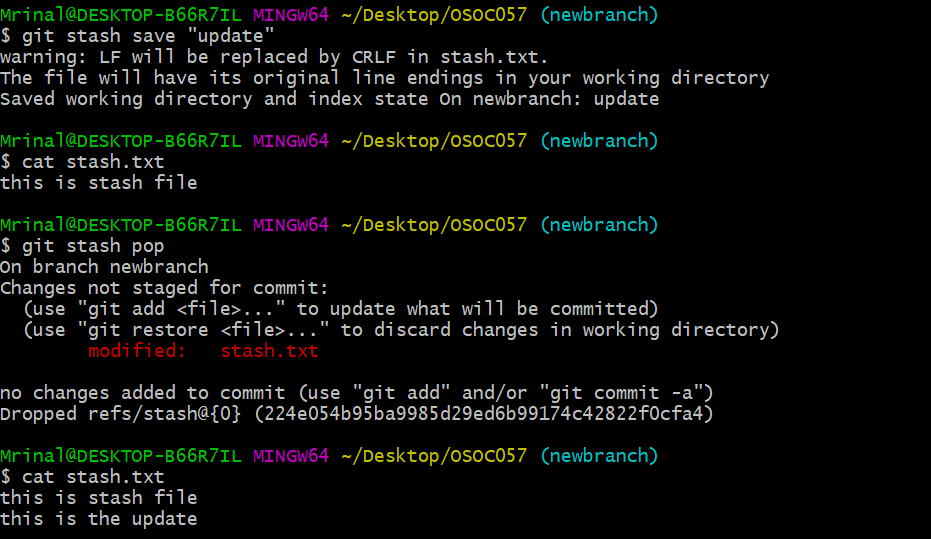
**Command:** git stash save “filename”

**To check the stash list :** git stash list

**To get the changes from stash :** git stash pop







**Q14. How to show the commits on ex (branch Ankur) that are not on ex (branch Ruchita )? (write command with screenshot of command line).**

Ans:-We can see the commits by using **git log** command.

**Syntax:** git log branchname

To see the commits of any branch

**Syntax:**git log filename

To see the commits of particular file

