GIT Assignment 1

Initialize a new Git repository in a directory of your choice. Add a simple text file to the repository and make the first commit.

To initialize a new Git repository, add a text file, and make the first commit, follow these steps:

1. **Open your terminal or command prompt:** Navigate to the directory where you want to create your Git repository. You can use `cd` (change directory) command to move to the desired location.

   ```
   cd /path/to/your/directory
   ```

2. **Initialize a new Git repository:** Use the `git init` command to create a new Git repository in the current directory.

   ```
   git init
   ```

   This command initializes an empty Git repository in the current directory.

3. **Create a simple text file:** You can create a new text file using any text editor or using terminal commands. Here's an example using command line:

   ```
   echo "Hello, Git!" > hello.txt
   ```

   This command creates a file named `hello.txt` with the content "Hello, Git!".

4. **Add the file to the Git repository:** Use `git add` to add `hello.txt` to the staging area (index). This prepares the file to be committed.

   ```
   git add hello.txt
   ```

   If you have multiple files or want to add all files in the directory, you can use `git add .` instead.

5. **Commit the changes:** Now, commit the changes to the Git repository using `git commit`. This creates a new commit with the files you've added to the staging area.

   ```
   git commit -m "Initial commit"
   ```

   Here, `-m` allows you to specify a commit message in quotes. The commit message should be concise but descriptive, summarizing the changes made in this commit.

6. **Verify the commit:** After committing, you can verify that everything is set up correctly by checking the status of the repository:

   ```
   git status
   ```

This command will show you the current status of the repository, confirming that there are no uncommitted changes.

Now, you've successfully initialized a new Git repository, added a simple text file (hello.txt), and made your first commit with the message "Initial commit".

Assignment 2: Branch Creation and Switching

Create a new branch named 'feature' and switch to it. Make changes in the 'feature' branch and commit them.

To create a new branch named 'feature', switch to it, make changes, and commit them, follow these steps:

Steps:

1. **Navigate to the Git repository:** Open your terminal or command prompt and change directory to your Git repository.

   ```
   cd /path/to/your/repository
   ```

2. **Create a new branch named 'feature':** Use the following command to create a new branch named 'feature'. This command creates the branch without switching to it.

   ```
   git branch feature
   ```

3. **Switch to the 'feature' branch:** To switch to the newly created 'feature' branch, use the git checkout command.

   ```
   git checkout feature
   ```

   Now, you are on the 'feature' branch and any changes you make will be within this branch.

4. **Make changes in the 'feature' branch:** Modify existing files or create new files as required for your feature development.

   Example: Add a new line to hello.txt.

   ```
   echo "This is a new feature!" >> hello.txt
   ```

5. **Stage and commit your changes:** Stage the changes to be committed and then commit them to the 'feature' branch.

   ```
   git add hello.txt    # Stage the changes to hello.txt
   git commit -m "Added a new feature message to hello.txt"
   ```

Ensure the commit message describes the changes made in the 'feature' branch.

6. **Verify your commit:** Check the status of your repository to verify that there are no uncommitted changes.

   git status

   This command will confirm the current status of your repository.

You have successfully created a new branch named 'feature', switched to it, made changes to hello.txt within the 'feature' branch, and committed those changes. Branching in Git allows for isolated development and experimentation, keeping main development lines clean until features are ready for integration.

LINUX ASSIGNMENT

1 Ensure the script check if specific file (e.g. myfile.txt) exists in current directory. If it exists, print "File exists", otherwise print "File not found".

Script:

```
#!/bin/bash

read -p "Enter the file name:" filename

if[ -f $filename]

then

        echo "File found"

else

        echo "File not found"

fi
```

2 Write a script that reads number from user until they enter '0'. The script should also print whether each number is odd or even.

Script:

```
#!/bin/bash

echo "Enter the number:"

read num

if((num%2==1)); then
```

```
        echo “odd”
else
        echo “even”
fi
```

3 Count the number of directories and files in specific folder.

Script:

```
#!/bin/bash
dir_count=0
file_count=0
folder=“/path/to/your/programs”
for item in “$folder”/*;do
     if[-d “$item”]; then
         dir_count=$((dir_count+1))
     else[-f “$item”];then
         file_count=$((file_count+1))
    fi
done
echo “Total number of directories: $dir_count”
echo “Total number of files: $file_count”
```

4 Find Smallest number from the Array

Script:

```
#!/bin/bash
numbers=(23 9 4 15 6 32)
smallest =${numbers[0]}
for num in “${number[@]}”;do
    if((num<smallest)); then
         smallest = $num
```

```
    fi
done
echo “The smallest number is: $smallest”
```

5 Find Sum of Array

Script:

```
#!/bin/bash
array=(5 7 9 3 6)
sum=0
for num in “${array[@]}”
do
    sum=$((sum + num))
done
echo “The sum of array is: $sum”
```

6 Display all the Directory Names

Script:

```
#!/bin/bash
# Display all the directory names in a specified folder
read -p “Enter the folder path:” folder
if [-d “$folder”]; then
        echo “Directories in $folder:”
        find “$folder” -type d
else
        echo “Folder not found”
fi
```

7 Check whether the Number is Palindrome or Not

Script:

```
#!/bin/bash
#check whether the number is palindrome or not
read -p “Enter a number:” number
reverse =$((echo “$number” | rev)
if[“$number” -eq “$reverse”]; then
        echo “$number is palindrome”
else
        echo “$number is not palindrome”
fi
```