Name: Vaishnavi Nagwekar
Batch : T13
Roll No: 66

EXPERIMENT NO: 03

Aim: To Perform various GIT operations on local and Remote repositories.

Theory :

Directory and File Commands

- **mkdir git**
  Creates a new directory (folder) named "git" in the current working directory. This command is used to create a new directory in Unix-like operating systems.
- **cd git**
  Changes the current working directory to the directory named "git." After executing this command, all subsequent commands or file operations will occur within the "git" directory.
  Note: "cd" stands for "change directory."
- **nano index.html**
  Opens the Nano text editor for the file named "index.html." Nano is a simple command-line text editor that allows you to view and edit files directly in the terminal.
- **touch teststatus**

  Creates an empty file named "teststatus" in the current directory. The touch command is commonly used to update the timestamps of a file or create an empty file if it doesn't exist.
- **git checkout -- teststatus**
  Discards changes to the file "teststatus" in the working directory. This reverts the file to the state it has in the last commit.

Git Configuration

- **git config --global user.name "Your Name"**
  Sets your global Git username, which will be associated with your commits.
- **git config --global user.email "youremail@example.com"**
  Sets your global Git email address, which will be associated with your commits.
- **git config --list**
  Displays the current configuration settings for Git, including user details and other preferences.

Staging and Committing Changes

- **git add <file>**
  Stages changes in the working directory for the next commit in Git. It prepares modifications, additions, or deletions to be included in the upcoming commit.
- **git commit -am "commit message"**
  Stages and commits all changes in tracked files with a commit message in a single command. This is a shorthand for git add <file> followed by git commit.
- **git log**
  Displays the commit history of the Git repository, showing a chronological list of commits, including commit hashes, author information, timestamps, and commit messages.
- **git log --oneline**
  Displays a simplified, one-line representation of the commit history, showing only the commit SHA-1 hash and the commit message.

Git Remote Operations

- **git clone <repository>**
  Creates a copy of a Git repository. This command duplicates the entire repository (files, commit history, branches) and downloads it to your local machine. It is often the initial step when working with a project hosted on a remote Git repository.
- **git pull**
  Fetches and integrates changes from a remote repository into the current branch of your local repository. It combines two actions: it fetches changes from the remote repository and merges them into your local branch.
- **git push**
  Uploads or pushes local changes from your Git repository to a remote repository. It updates the remote repository with the latest changes made in your local branch.
- **git fetch**
  Retrieves changes from a remote repository, including new branches or changes made since your last interaction. However, it does not automatically merge these changes into your local branches. After using git fetch, you can inspect the changes and decide whether to integrate them using git merge or git rebase.

```
Lab203@203-003 MINGW64 ~/git-dvcs43 (master)
$ git config --global
usage: git config [<options>]

Config file location
    --[no-]global         use global config file
    --[no-]system         use system config file
    --[no-]local          use repository config file
    --[no-]worktree       use per-worktree config file
    -f, --[no-]file <file>
                          use given config file
    --[no-]blob <blob-id> read config from given blob object

Action
    --[no-]get            get value: name [value-pattern]
    --[no-]get-all        get all values: key [value-pattern]
    --[no-]get-regexp     get values for regexp: name-regex [value-pattern]
    --[no-]get-urlmatch   get value specific for the URL: section[.var] URL
    --[no-]replace-all    replace all matching variables: name value [value-patt
ern]
    --[no-]add            add a new variable: name value
    --[no-]unset          remove a variable: name [value-pattern]
    --[no-]unset-all      remove all matches: name [value-pattern]
    --[no-]rename-section rename section: old-name new-name
    --[no-]remove-section remove a section: name
    -l, --[no-]list       list all
    --[no-]fixed-value    use string equality when comparing values to 'value-pa
ttern'
    -e, --[no-]edit       open an editor
    --[no-]get-color      find the color configured: slot [default]
    --[no-]get-colorbool  find the color setting: slot [stdout-is-tty]

Type
    -t, --[no-]type <type>
                          value is given this type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --bool-or-str         value is --bool or string
    --path                value is a path (file or directory name)
    --expiry-date         value is an expiry date

Other
    -z, --[no-]null       terminate values with NUL byte
    --[no-]name-only      show variable names only
    --[no-]includes       respect include directives on lookup
    --[no-]show-origin    show origin of config (file, standard input, blob, com
mand line)
    --[no-]show-scope     show scope of config (worktree, local, global, system,
 command)
    --[no-]default <value>
                          with --get, use default value when missing entry


Lab203@203-003 MINGW64 ~/git-dvcs43 (master)
$ git config --global user.name "meetk"

Lab203@203-003 MINGW64 ~/git-dvcs43 (master)
$ ^C

Lab203@203-003 MINGW64 ~/git-dvcs43 (master)
$ git config --global user.email "meetkadam1812@gmail.com"

Lab203@203-003 MINGW64 ~/git-dvcs43 (master)
$ git config --global --list
user.name=meetk
user.email=meetkadam1812@gmail.com
```

```
15L@2O3-005 MINGW64 ~/git-dvcs (master)
$  mkdir git-demo-project

15L@2O3-005 MINGW64 ~/git-dvcs (master)
$ cd git-demo-project/

15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$  git init
Initialized empty Git repository in C:/Users/15L/git-dvcs/git-demo-project/.git/
```

```
15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ ls -a
./  ../  .git/

15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ ls -al
total 4
drwxr-xr-x 1 15L 197121 0 Feb 12 11:10 ./
drwxr-xr-x 1 15L 197121 0 Feb 12 11:10 ../
drwxr-xr-x 1 15L 197121 0 Feb 12 11:10 .git/
```

```
15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git add .

15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
```

```
15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git commit -m "First Commit"
[master (root-commit) 54ca444] First Commit
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
```

```
15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.htm

nothing added to commit but untracked files present (use "git add" to track)
```

```
15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git add index.htm

15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.htm
```

```
15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git commit -am "Express commit"
[master 9404a12] Express commit
 1 file changed, 1 insertion(+)
 create mode 100644 index.htm

15L@2O3-005 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Conclusion : Successfully performed various GIT operations on local and Remote repositories.