1. A. Write the steps for Git installation & setup.
Download & install Git on your system.

---

🔷 **Step 1: Download Git**

1. Open your browser and go to git-scm.com.
2. On the homepage, you'll see download options for Windows, Mac, and Linux.
3. Click the one that matches your system (most people use Windows).
4. The download will start automatically.

---

🔷 **Step 2: Install Git**

1. Once the file is downloaded, double-click it to open.
2. The installer will open with multiple steps. Don't panic—just keep clicking **Next** unless you want something very specific.
3. Make sure the option **"Add Git to PATH"** is selected. (This lets you use Git from the Command Prompt or Terminal.)
4. Finish the installation by clicking **Install**, then **Finish**.

✅ Now Git is installed on your computer.

---

🔷 **Step 3: Verify Installation**

1. Open **Command Prompt** (Windows) or **Terminal** (Mac/Linux).
2. Type:

```
git --version
```

3. If you see something like git version 2.45.1, it means Git is installed correctly.

---

🔷 **Step 4: Git Setup (First-time Configuration)**

Before using Git, you need to tell it who you are (so your work is marked with your name/email).

1. In your **Command Prompt/Terminal**, type:

   ```
   git config --global user.name "Your Name"

   git config --global user.email "your@email.com"
   ```

Example:

```
git config --global user.name "user123"

git config --global user.email "user123@example.com"
```

2. To check if it worked, type:

   ```
   git config --list
   ```

You'll see your name and email listed.

---

🔷 **Step 5: Test Git with a Folder**

1. Create a new folder on your Desktop (example: MyProject).
2. Right-click inside the folder → choose **Git Bash Here** (on Windows) or just open terminal in that folder.
3. Run:

   ```
   git init
   ```

This sets up Git inside that folder. Now Git will track any changes you make.

---

🔷 **Step 6: First File Commit**

1. Inside that folder, create a simple text file like notes.txt and type anything inside.
2. Run:

   ```
   git add notes.txt

   git commit -m "My first commit"
   ```

✔ Congrats! You just made your first snapshot (commit) with Git.

---

📌 **Explanation in Simple Words**

- **Install Git** → Like installing WhatsApp on your phone.
- **Setup name & email** → Like setting up your WhatsApp profile, so everyone knows who you are.
- **Init (git init)** → Like creating a new WhatsApp group, but here it's for your code.
- **Add & Commit** → Like sending your first message in the group chat — Git remembers it forever.

---

## B. Perform the following operations using local Git

### i. Creating a Repo

### ii. Cloning a Repo

### iii. Making changes , staging & commiting changes to the files

### iv. Perform Git Branching & Merging operations

### v. Checks the logs & history & version of the files

---

🔷 **i) Creating a Repo**

**Repo = Repository = Project folder tracked by Git.**
It's like starting a diary where Git will remember every change you write.

**Steps:**

1. Create a folder on your computer: MyProject.
2. Open **Terminal/Command Prompt** in that folder.
3. Run:

```
git init
```

✅ Now this folder is a Git repository. Git is watching this folder and is ready to track changes.

---

## 🔷 ii) Cloning a Repository

**Cloning = Downloading someone else's Git project (repo) onto your computer.**
Think of it like making a copy of your friend's diary so you can also write in it.

**Steps:**

1. Suppose the repo link is:

   https://github.com/example/repo.git

2. Run:

   ```
   git clone https://github.com/example/repo.git
   ```

3. This creates a new folder on your computer with all the project files.

---

## 🔷 iii) Making Changes, Staging & Committing

Now you already have a repo. Let's change something.

**Example:**

1. Open a file in your local PC(say notes.txt) and write:

   Hello Git

2. Now tell Git that this file is ready to be saved (this is called **staging**):

   ```
   git add notes.txt
   ```

3. Save this change with a message (**commit**):

   ```
   git commit -m "Added hello message to notes.txt"
   ```

4. Uploading the files which are added and commited (**push**):

   ```
   git push -u origin main
   ```

✅ Git takes a snapshot of your file at this point. If you mess up later, you can always roll back.

👉 In simple terms:

- **Edit file** → Write your diary entry.

- **Staging (git add)** → Mark which page you want Git to remember.
- **Commit (git commit)** → Git locks that page with a timestamp + your name.
- **Push (git push -u origin main)** → Now Git uploads the files which are added and committed

---

## 🔷 iv) Git Branching & Merging

**Branch = A separate workspace to try new ideas.**
Think of your repo as a **tree**:

- Main tree = main branch
- Side branches = experiments.

**Steps:**

1. Create a new branch:

   ```
   git branch new-feature
   ```

2. Switch to that branch:

   ```
   git checkout new-feature
   ```

3. Make changes in files, then commit (same steps as before).
4. Once happy, merge it back into main:

   ```
   git checkout main

   git merge new-feature
   ```

✅ Now your new work is combined with the main project.

👉 Simple analogy:

- **Branch** = Creating a photocopy of your diary to test ideas.
- **Merge** = Taking the good parts from that photocopy and adding them back into your original diary.

---

### 🔷 v) Check Logs, History & File Versions

Git keeps a **history book** of everything you do.

**Steps:**

1. To see all commits:

```
git log
```

(Press q to quit the log view.)

2. To see a simpler one-line version:

```
git log --oneline
```

3. To see changes inside files:

```
git diff
```

4. To go back to an older version of a file:

```
git checkout <commit_id> filename
```

(Here <commit_id> comes from git log.)

👉 Analogy:

- **Log** = Table of contents in your diary showing when and what was written.
- **Diff** = See “before vs after” of your pages.
- **Checkout** = Jump into a past version of your diary.

---

### ✅ Summary (Non-Technical Explanation)

- **Repo** → Your project diary.
- **Clone** → Copy someone else’s diary.
- **Add & Commit** → Write + save a snapshot of your page.
- **Branch** → Make a duplicate diary for experiments.
- **Merge** → Combine experiment back into main diary.
- **Log/History** → Flip back through the pages of your diary anytime.

---