|  |  |  |
| --- | --- | --- |
| **S. No** | **Program Title** | **Page No.** |
| 1. | Add collaborators on GitHub Repo | 16 |
| 2. | Fork and Commit | 21 |
| 3. | Merge and Resolve conflicts created due to own activity and collaborators activity. | 24 |
| 4. | Reset and revert | 27 |

Experiment No. 06

**Aim:** Add collaborators on GitHub Repo

# Theory:

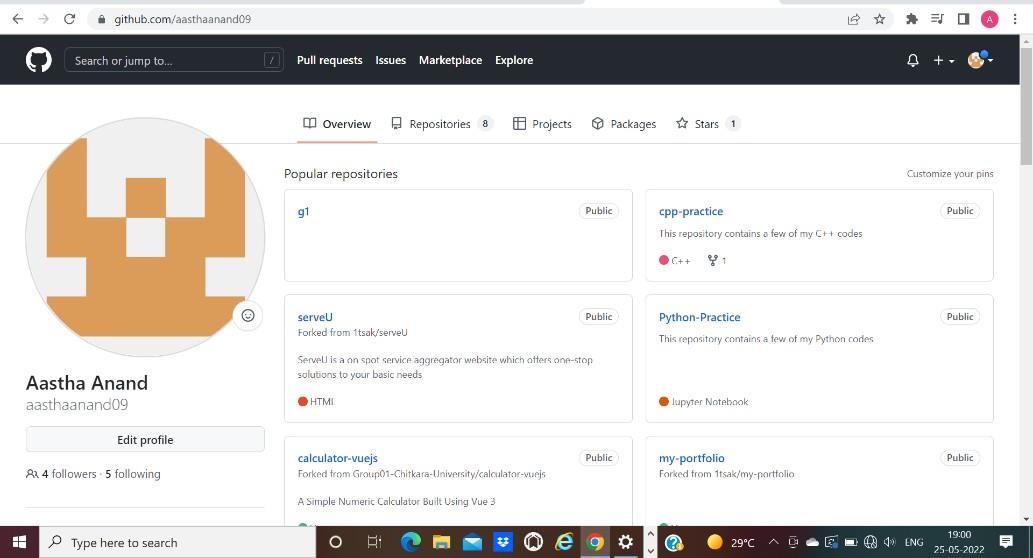
Whenever you make a repository in GitHub, not everyone has the permission to change or push codes into your repository. The users have a read-only access. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate to the project.

GitHub also restricts the number of collaborators we can invite within a period of 24 hours. If we exceed the limit, then either we have to wait for 24-hours or we can also create an organization to collaborate with more people.

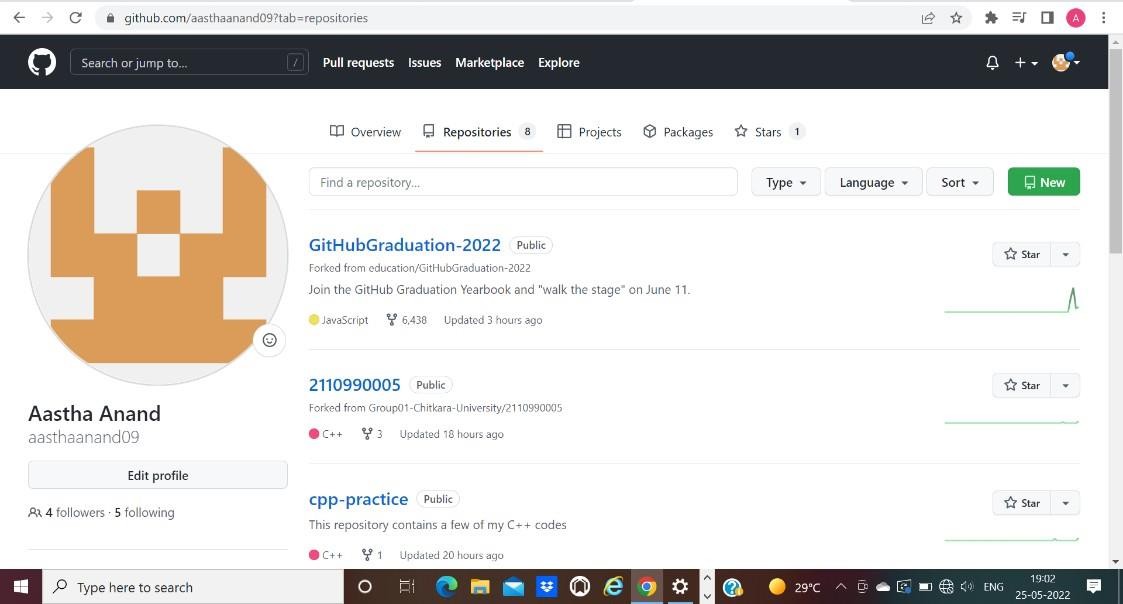
Being a collaborator, the user can create, merge and close pull requests in the repository. They can also remove them as the collaborator.

# Procedure:

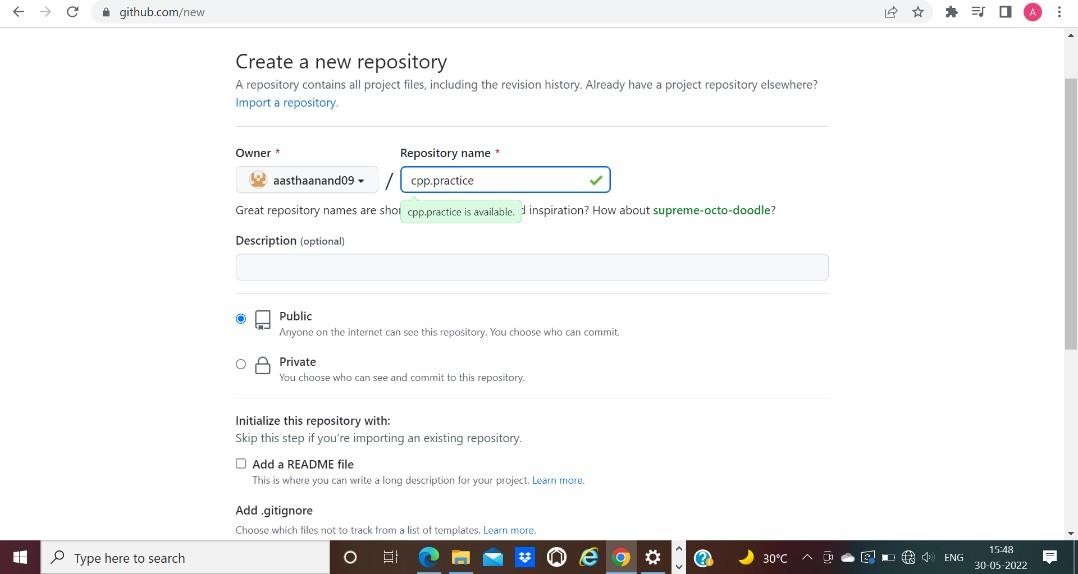
1. Login to your GitHub account and you will land on thehomepage as shown below. Click on Repositories option in the menu bar.



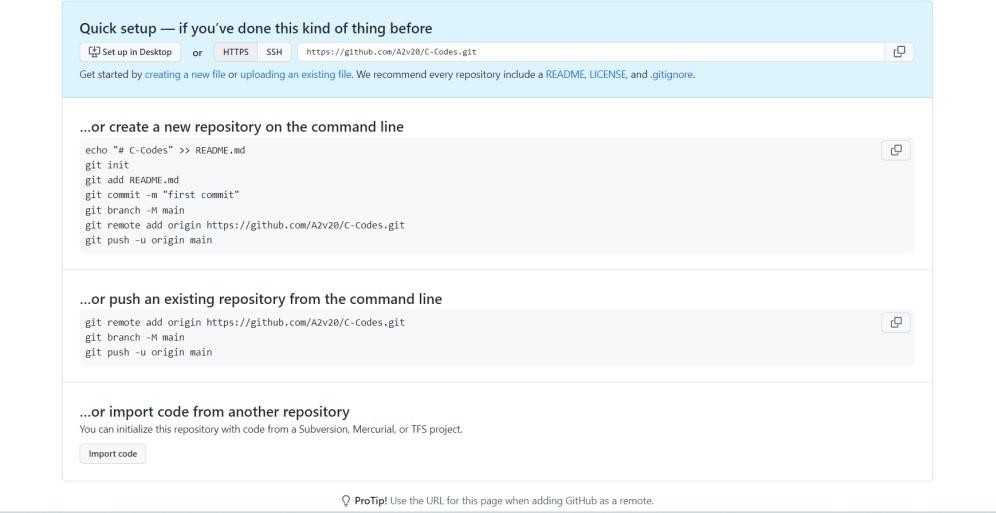
1. Click on the ‘New’ button in the top right corner.



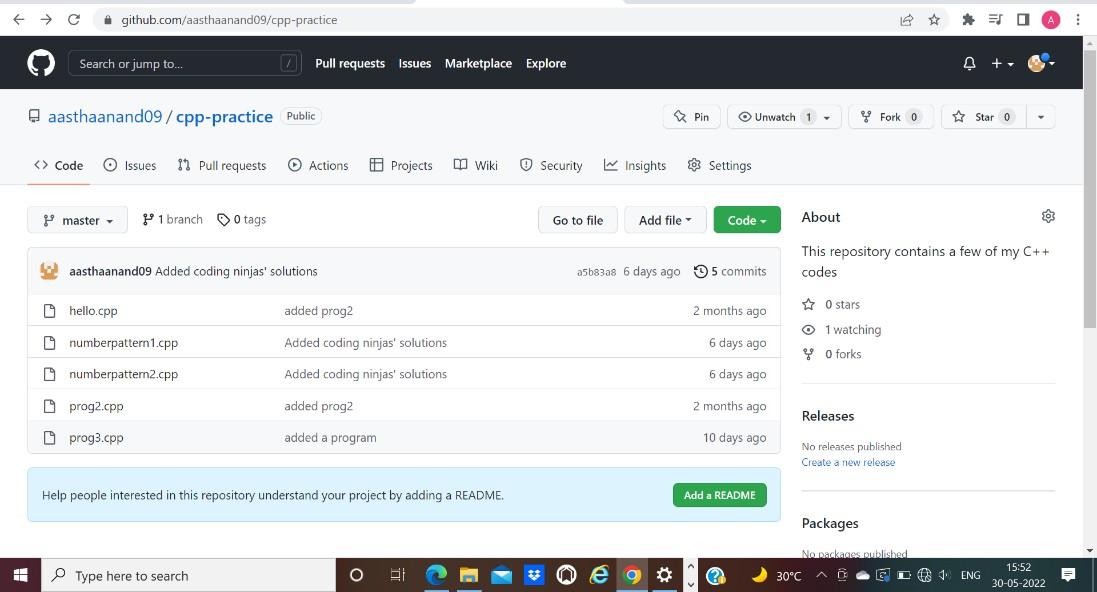
1. Enter the Repository name and add the description of therepository.
2. Select if you want the repository to be public or private.



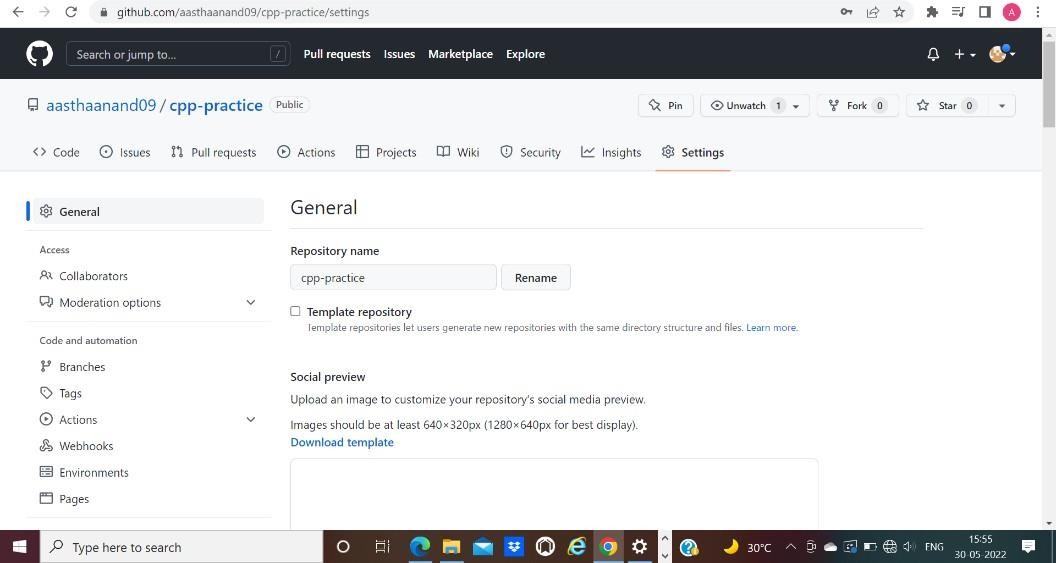
1. If you want to import code from an existing repository selectthe import code option.



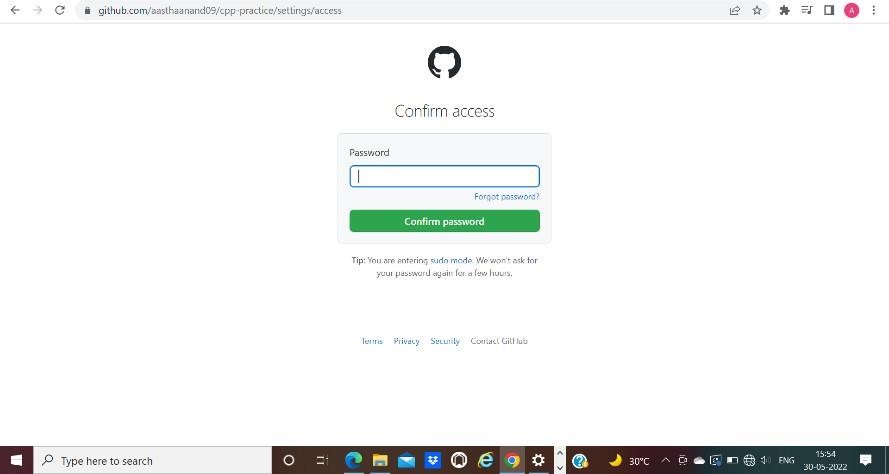
1. Now, you have created your repository successfully.
2. To add collaborators to your repository, open your repository and select settings option in the navigation bar.



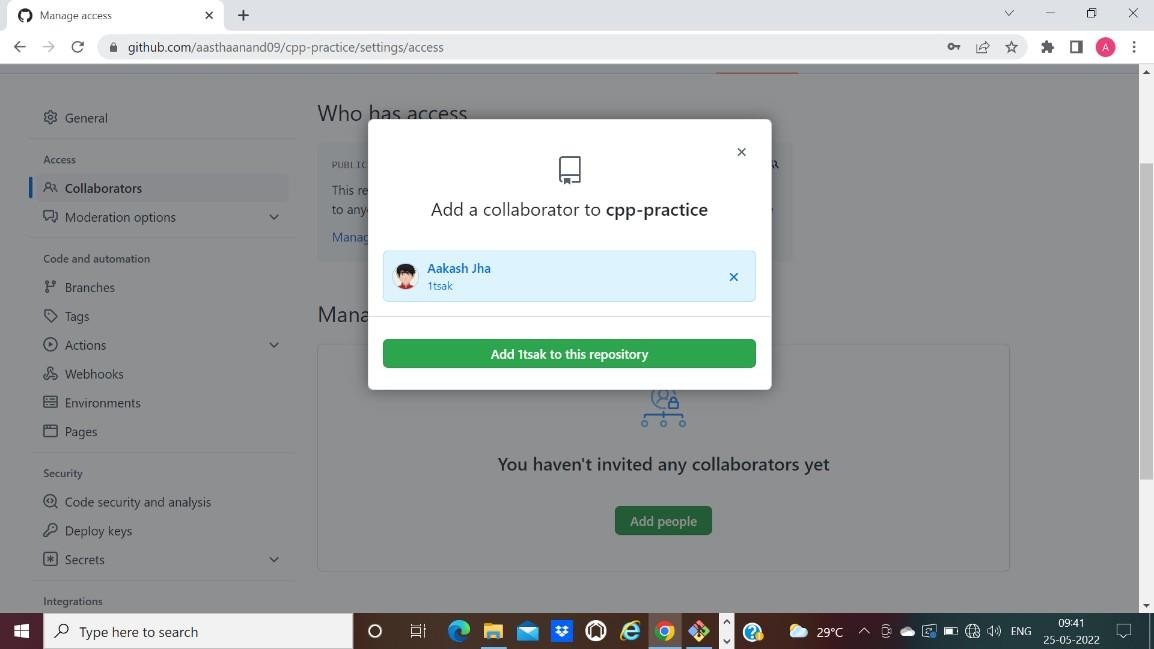
1. Click on Collaborators option under the access tab.



1. After clicking on collaborators, GitHub asks you to enter your password to confirm the access to the repository.

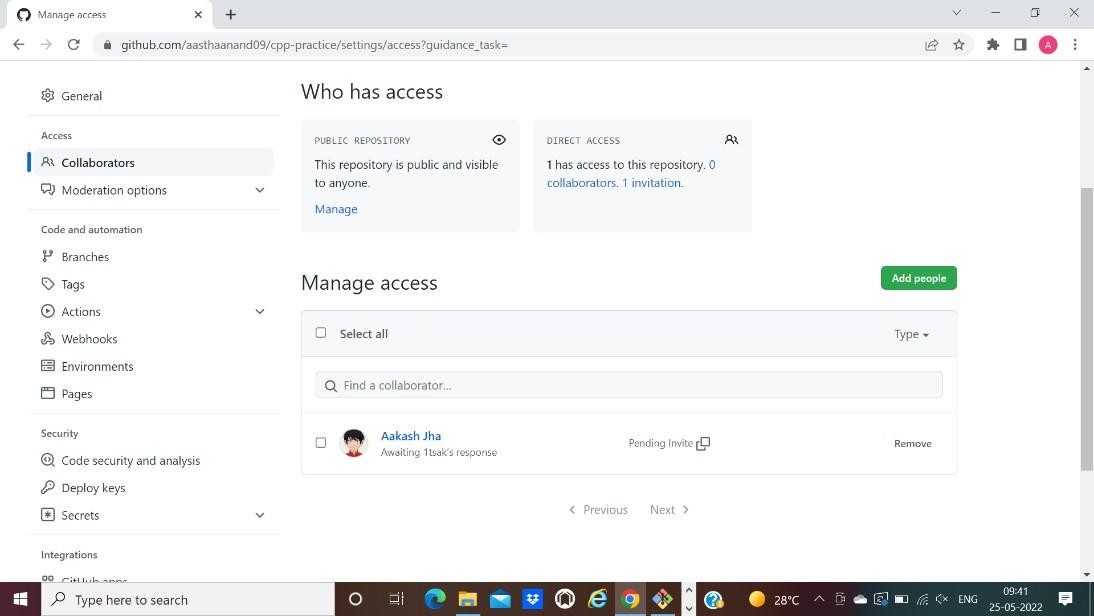


1. After entering the password, you can manage access andadd/remove team members to your project.
2. To add members, click on the add people option and search the id of your respective team member.



1. To remove any member, click on remove option

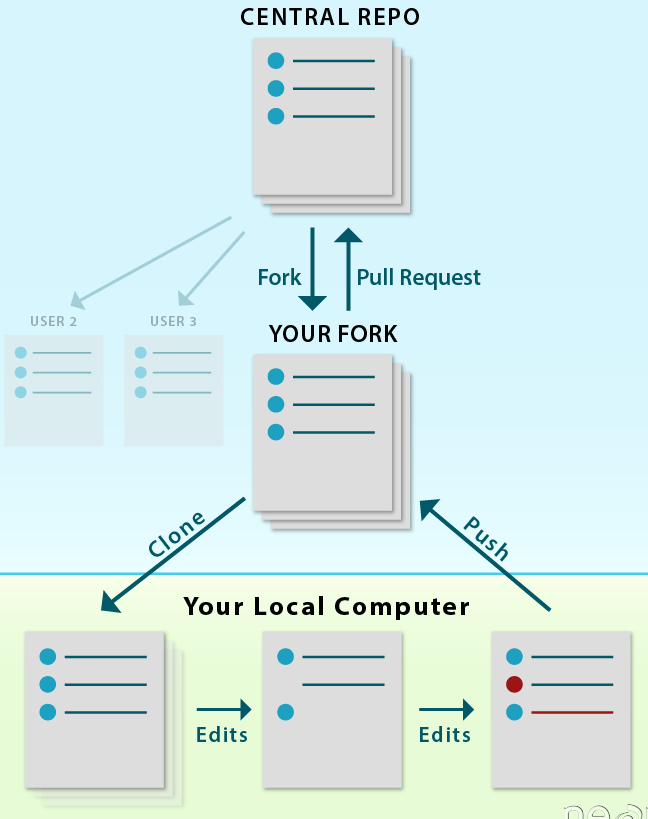
available in the last column of member’s respective row.



## Experiment No. 07

**Aim:** Fork and Commit

**Theory:** A fork is a copy of a repository that you manage. It allows us to freely experiment with the data. After creating a fork, we can make any desired change like adding collaborators, rename files, generate GitHub pages but all these changes won’t be reflected in the original repository.



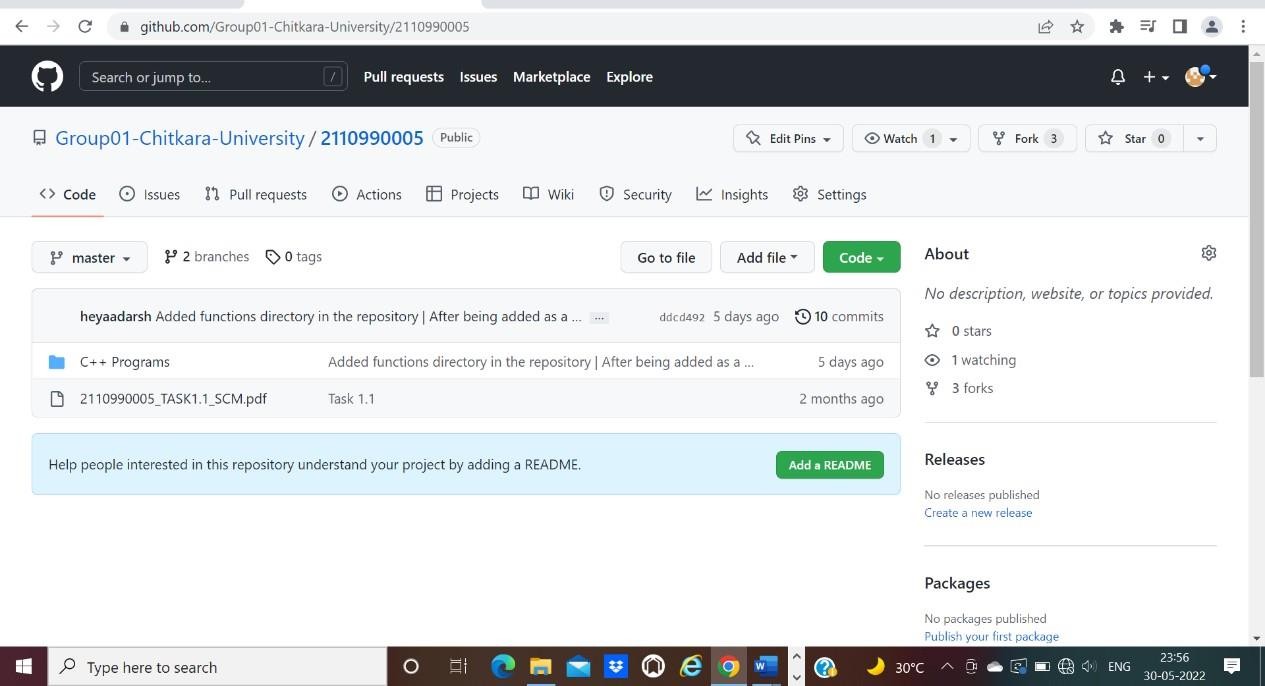
reference for picture: https://[www.earthdatascience.org](http://www.earthdatascience.org/)

To import the changes into the original repository, the user needs to send a pull request to the maintainer. If the maintainer closes the pull request only then the content can be added to the original repository.

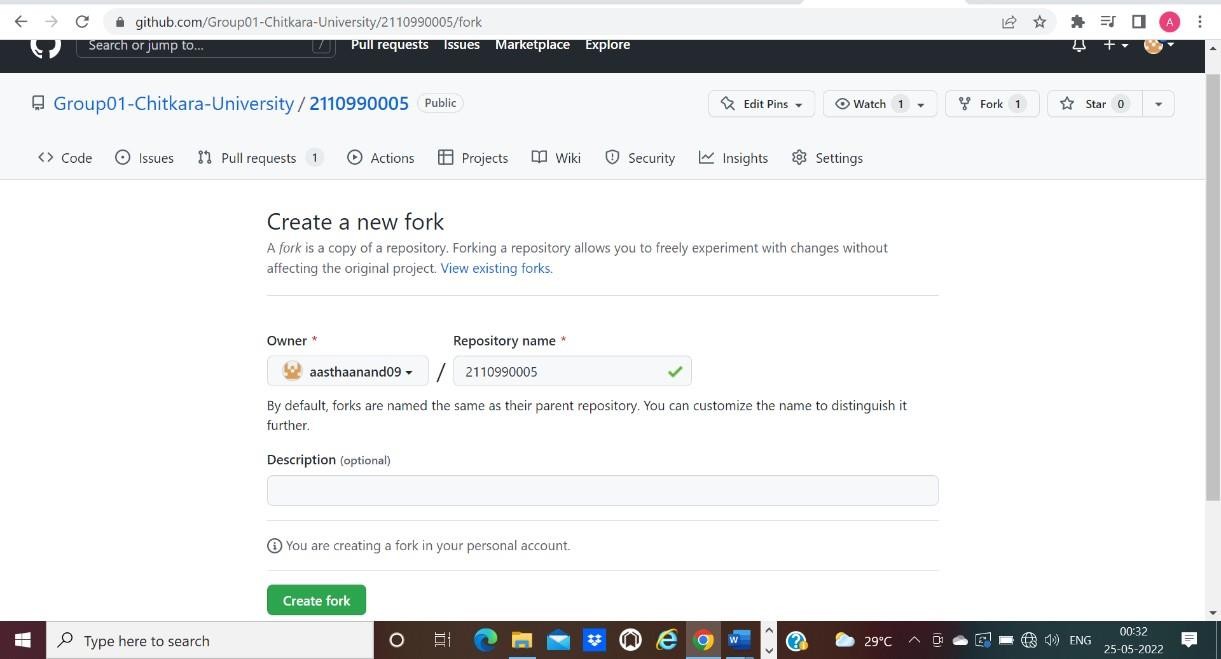
Forking is a better method than directly cloning any repository, as in cloning only the default branch is cloned whereas forking creates a clone of the complete repo and also allows us to push the changes to the main repository by using open and close pull request.

# Procedure:

* 1. To fork a repository first thing you need to do is, sign in to your GitHub account and then you come to the repository you want to fork, so here for demo purpose am using **Group01-Chitkara- University/2110990005** repository.

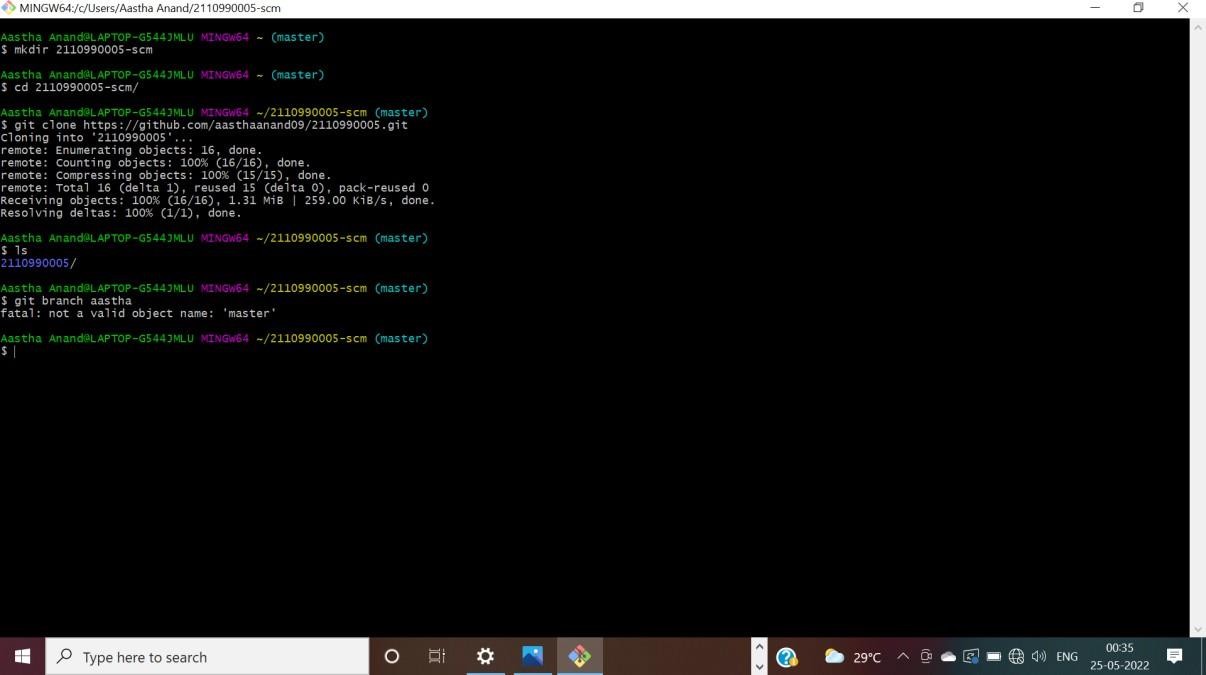


* 1. Click on the **Fork** button on right upside corner. Then it will ask to create a new fork, add description if you want and then click on create fork.

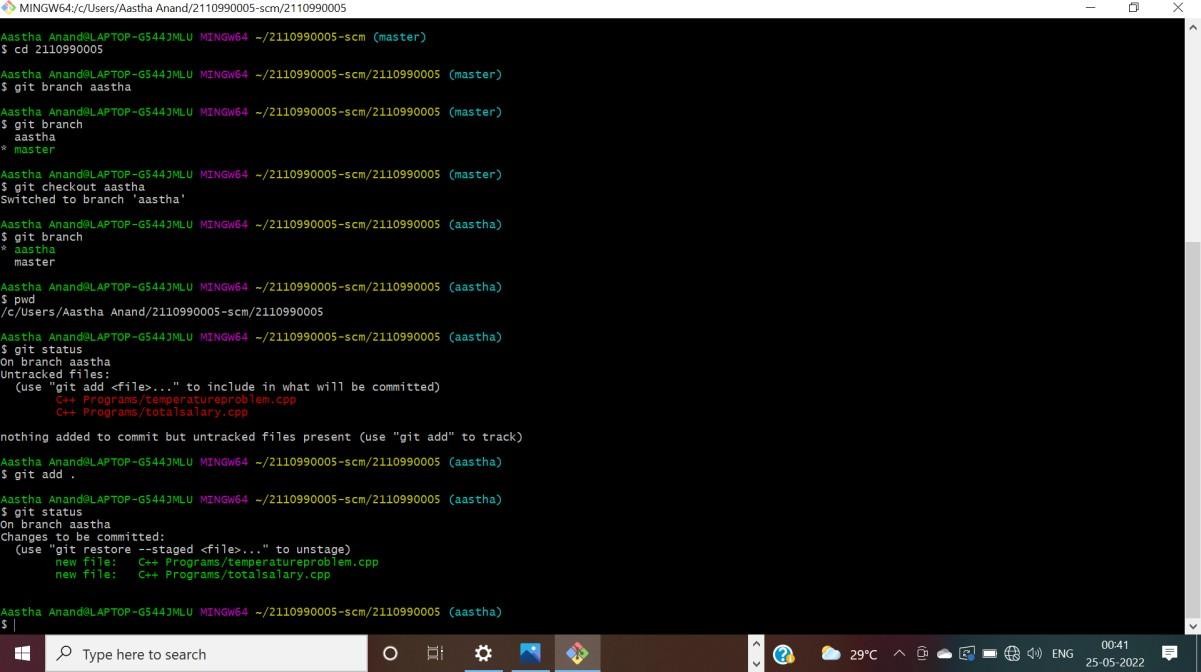


* 1. Now you will have a copy of the repo you have forked from other user. Now you can do any modification you want without making changes to main source code.
  2. Now type git clone [https://github.com/Group01-Chitkara-](https://github.com/Group01-Chitkara-University/2110990005.git) [University/2110990005.git](https://github.com/Group01-Chitkara-University/2110990005.git) on git.

Git clone <url> --> This command is used to fetch the remote repo or to clone the repo.



* 1. Now Open the file make changes in it and commit it and push it to remote.

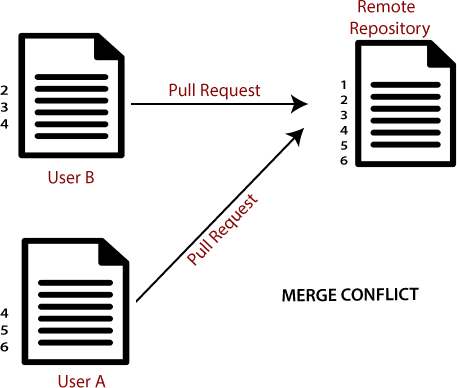


**Experiment No. 08**

**Aim:** Merge and Resolve conflicts created due to own activity and collaborators activity.

# Theory:

Version control systems are all about managing contributions between multiple distributed authors (usually developers). Sometimes multiple developers may try to edit the same content. If Developer A tries to edit code that Developer B is editing a conflict may occur.



reference for picture: https://[www.javatpoint.com/git-merge-and-merge-conflict](http://www.javatpoint.com/git-merge-and-merge-conflict)

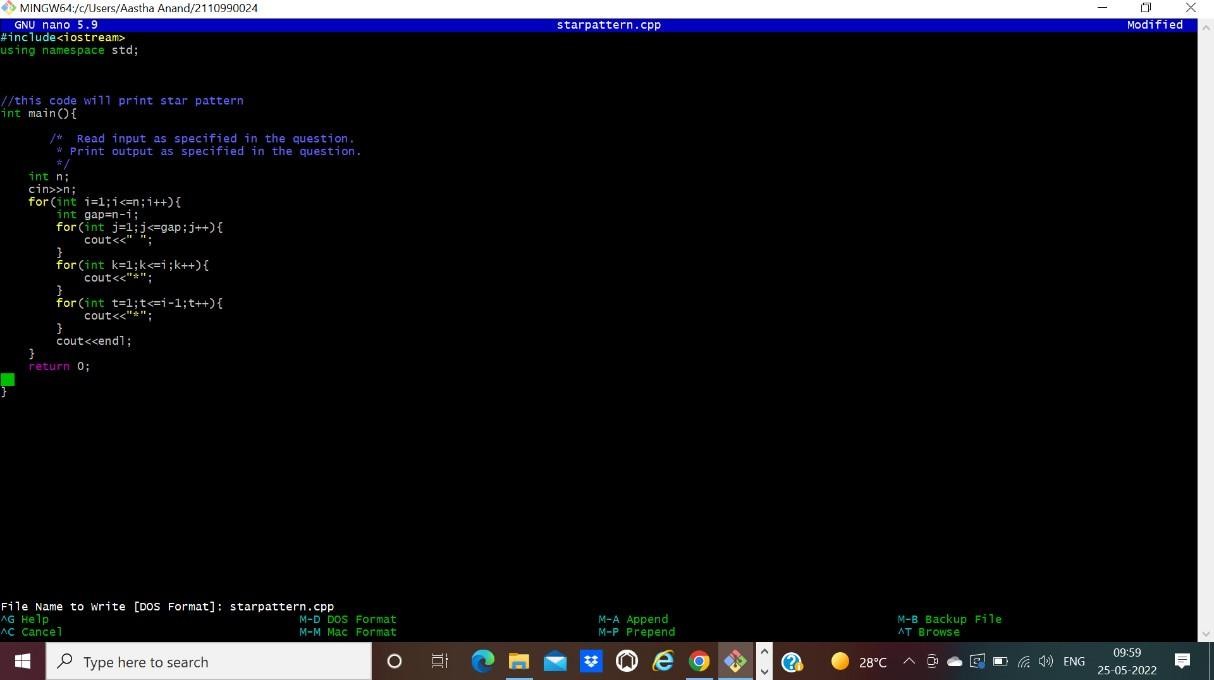
If you have a merge conflict on the command line, you cannot push your local changes to GitHub until you resolve the merge conflict locally on your computer.

To alleviate the occurrence of conflicts developers will work in

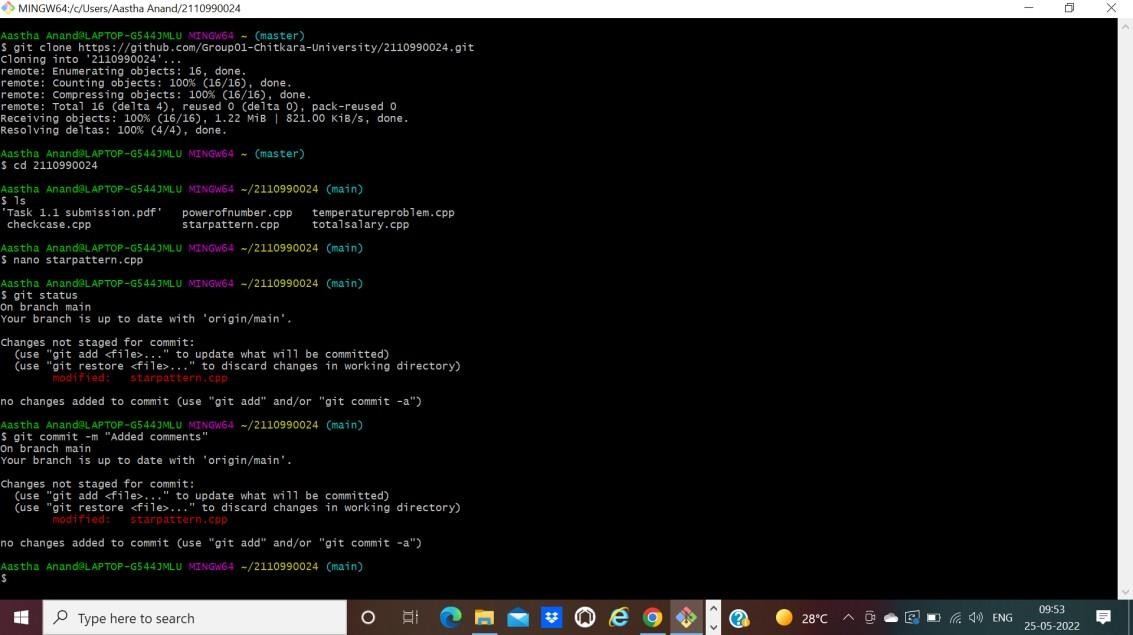
separate [isolated branches](https://www.atlassian.com/git/tutorials/using-branches). If a merge conflict still arises between the compare branch and base branch in your pull request, you can view a list of the files with conflicting changes above the Merge pull request button. The Merge pull request button is deactivated until you've resolved all conflicts between the compare branch and base branch.

# Procedure:

* + 1. Do changes in master branch and commit those change. And checkout to different branch and again do changes and commit it. Now checkout to master branch and merge that branch in master.

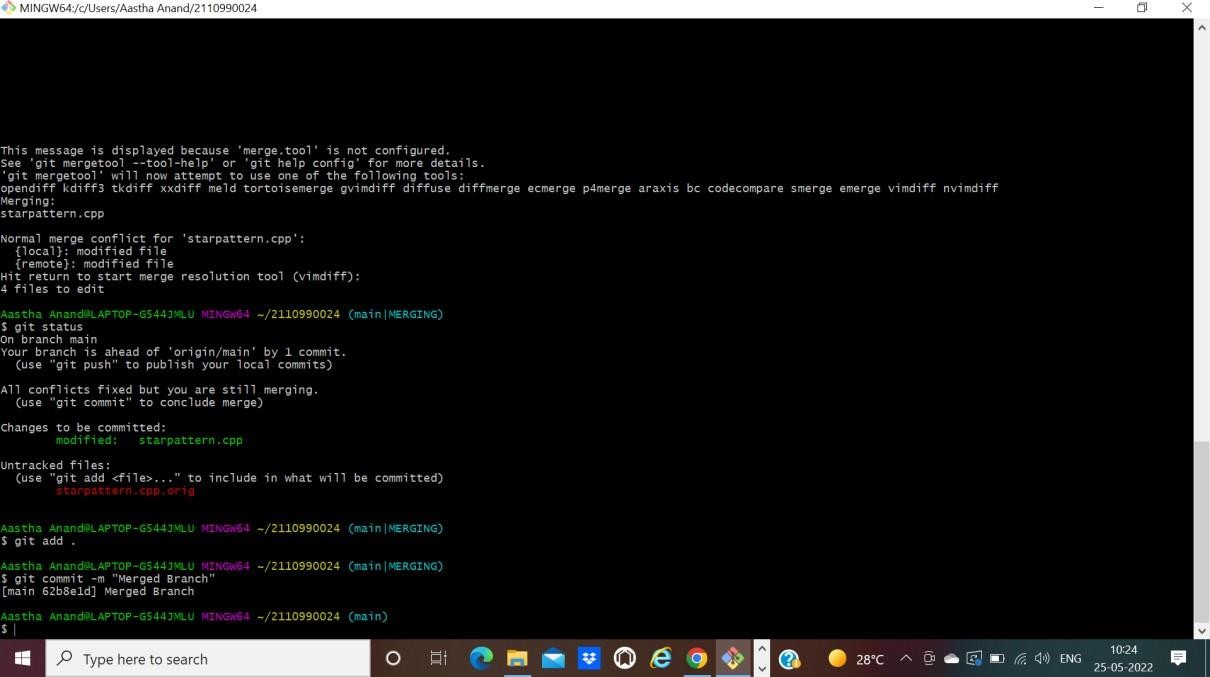


1. Now try to merge it will give Conflicts Error.

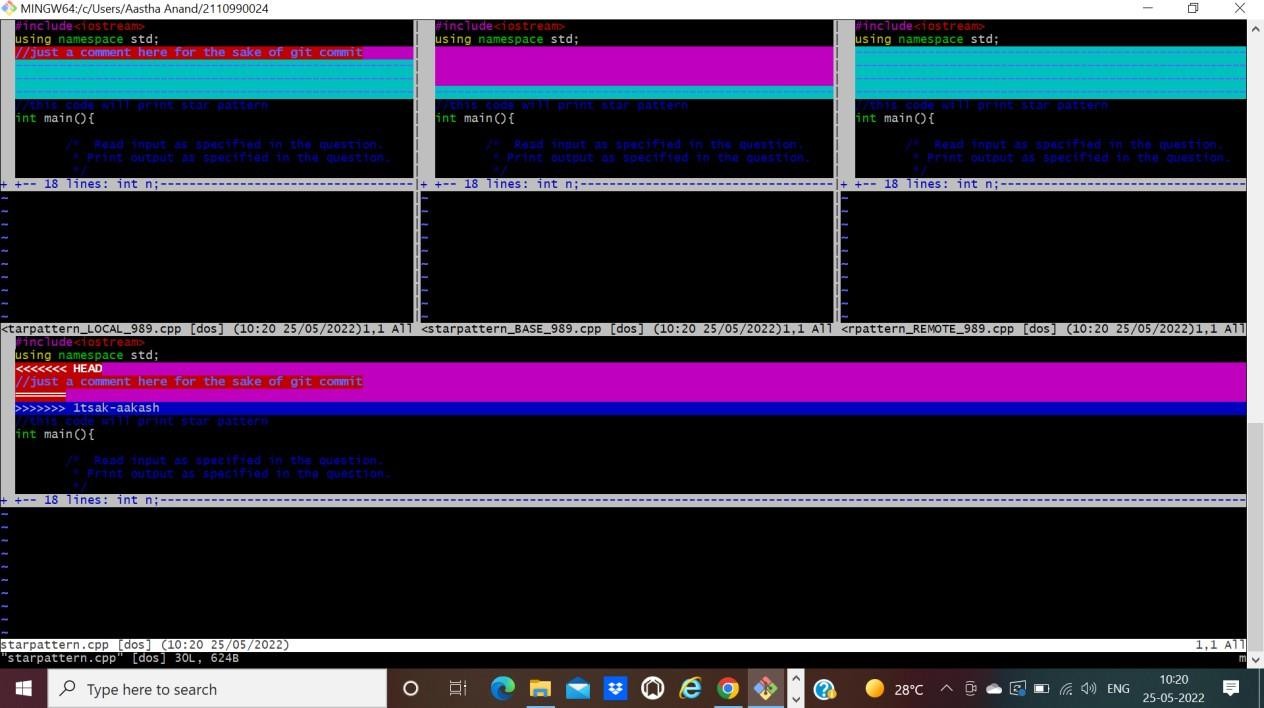


1. Use Command “git mergetool” to solve the conflict.

**git -mergetool** – Run merge conflict resolution tools to resolve merge conflicts.



1. Press “I” to insert, after insertion. Press “:wq”. The merge conflict is solved and aastha branch is merged to master branch.



## Experiment No. 09

**Aim:** Reset and Revert

# Theory:

Git-revert – Revert some existing commits.

A reset is an operation that takes a specified commit and resets the "three trees" to match the state of the repository at that specified commit. A reset can be invoked in three different modes which correspond to the three trees. In reset, rest of the commits wash out after the mentioned commit. This is a limitation of reset command that we cannot have any random access.

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.

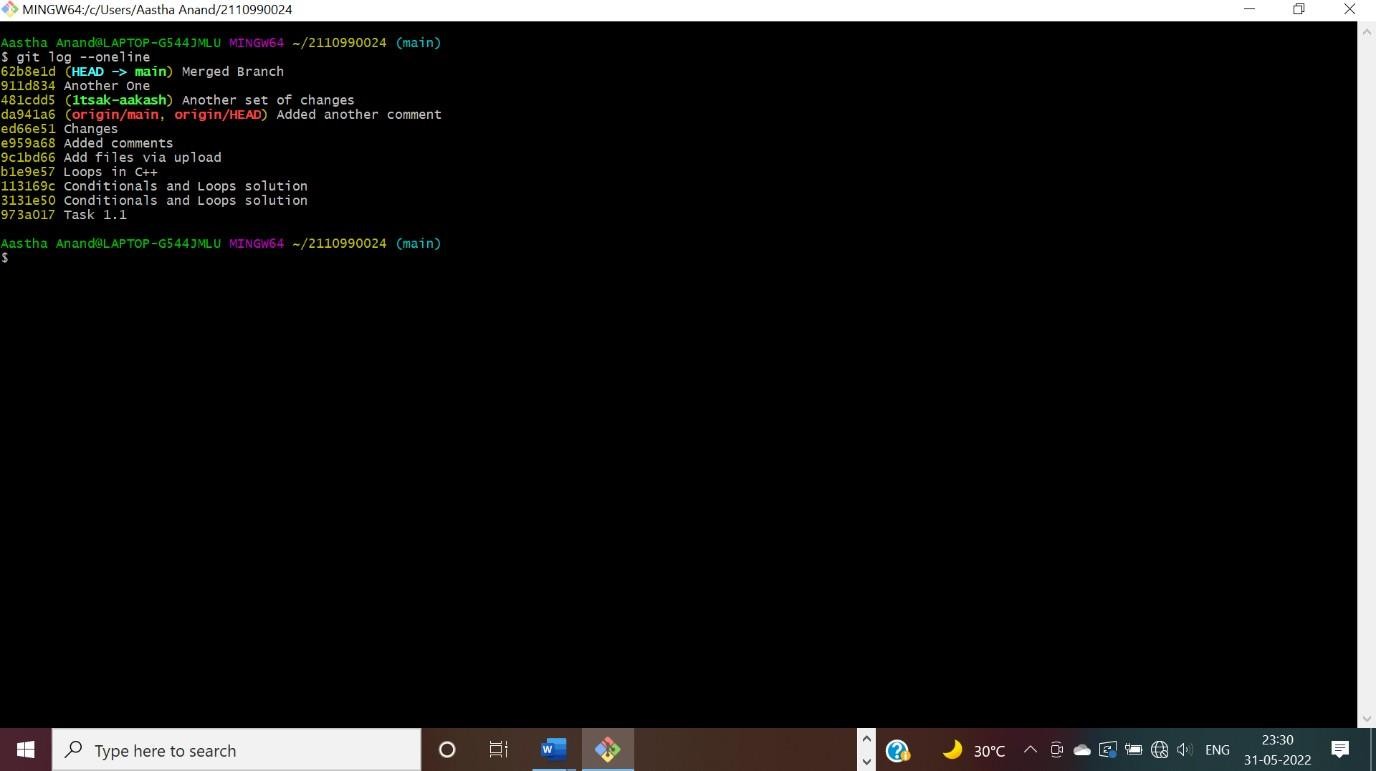
These two features justify the Version- controlled feature of the git as we can rollback to any version at any time.

# PROCEDURE:

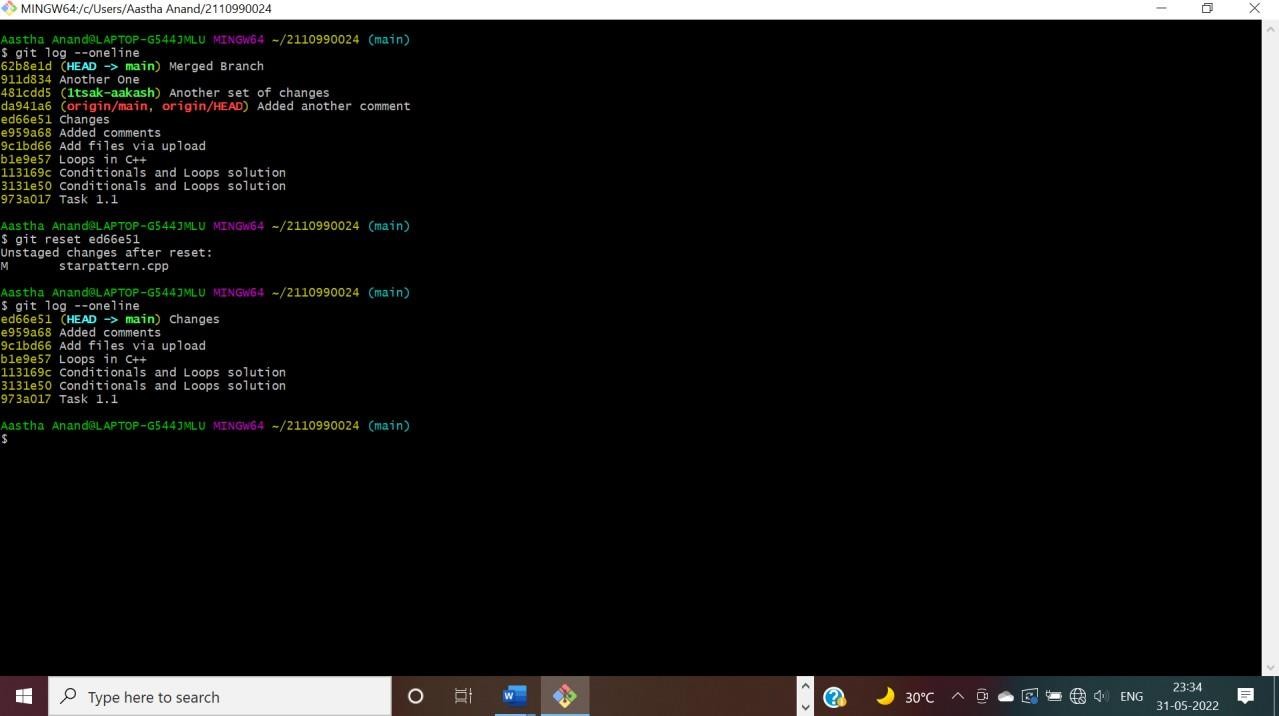
Firstly, prepare a log of multiple commits to make the reset and revert command function.

**Reset:** reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.

1. Create few files, stage them and commit.
2. Check the git log

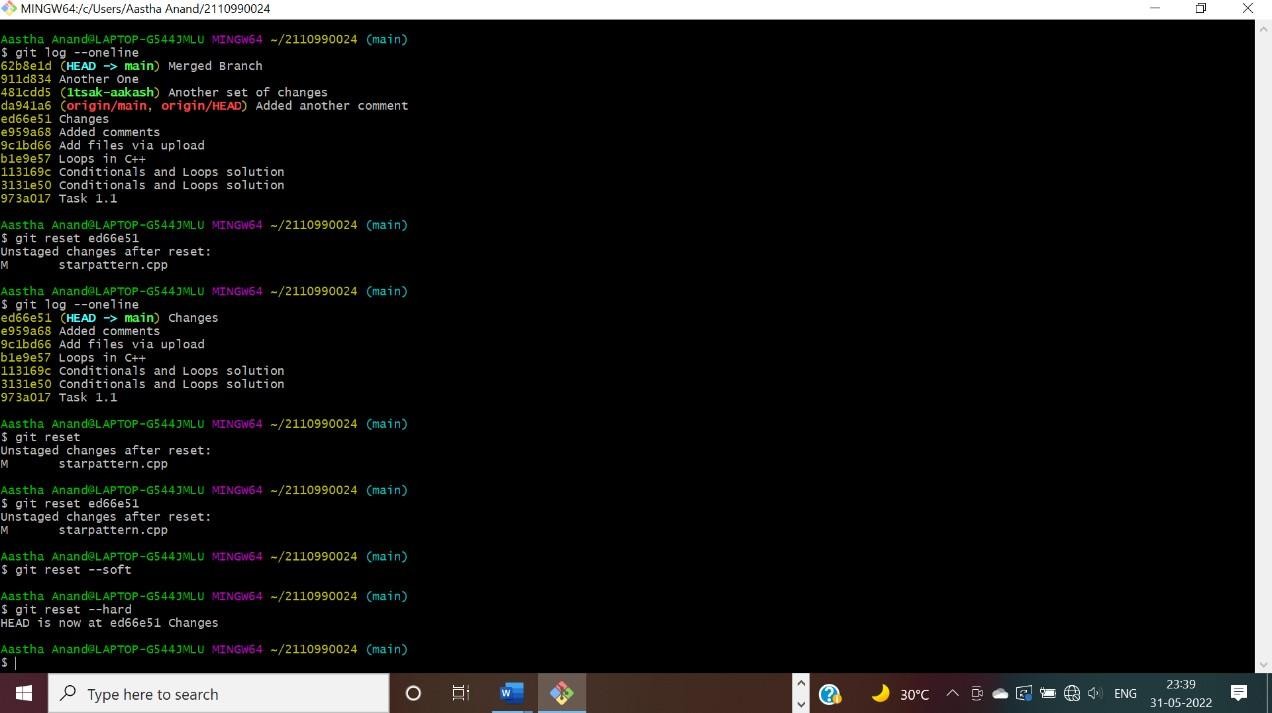


1. Pick any commit where you want the repository to rollback. Copy its checksum and paste it in the $ **git reset checksum** command.



The head is now pointing the commit whose checksum we have provided that means the commits that followed vanished.

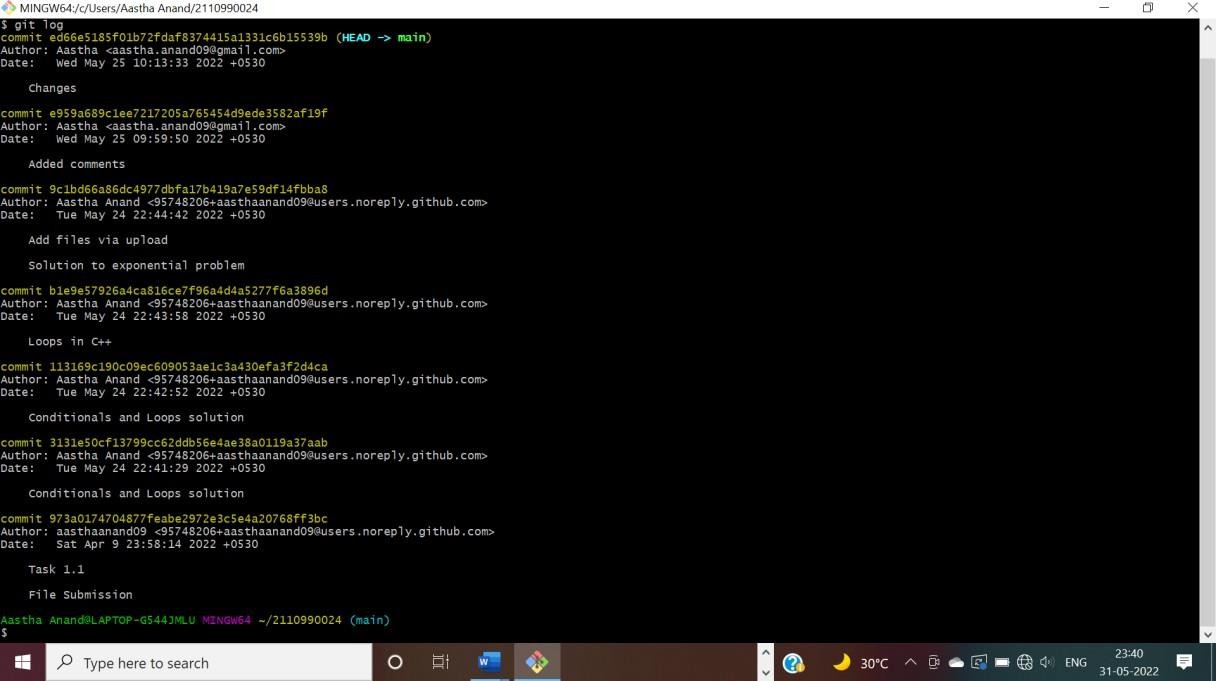
1. In you want undo this change, you copy the checksum of the commit you want back and run the same command again.



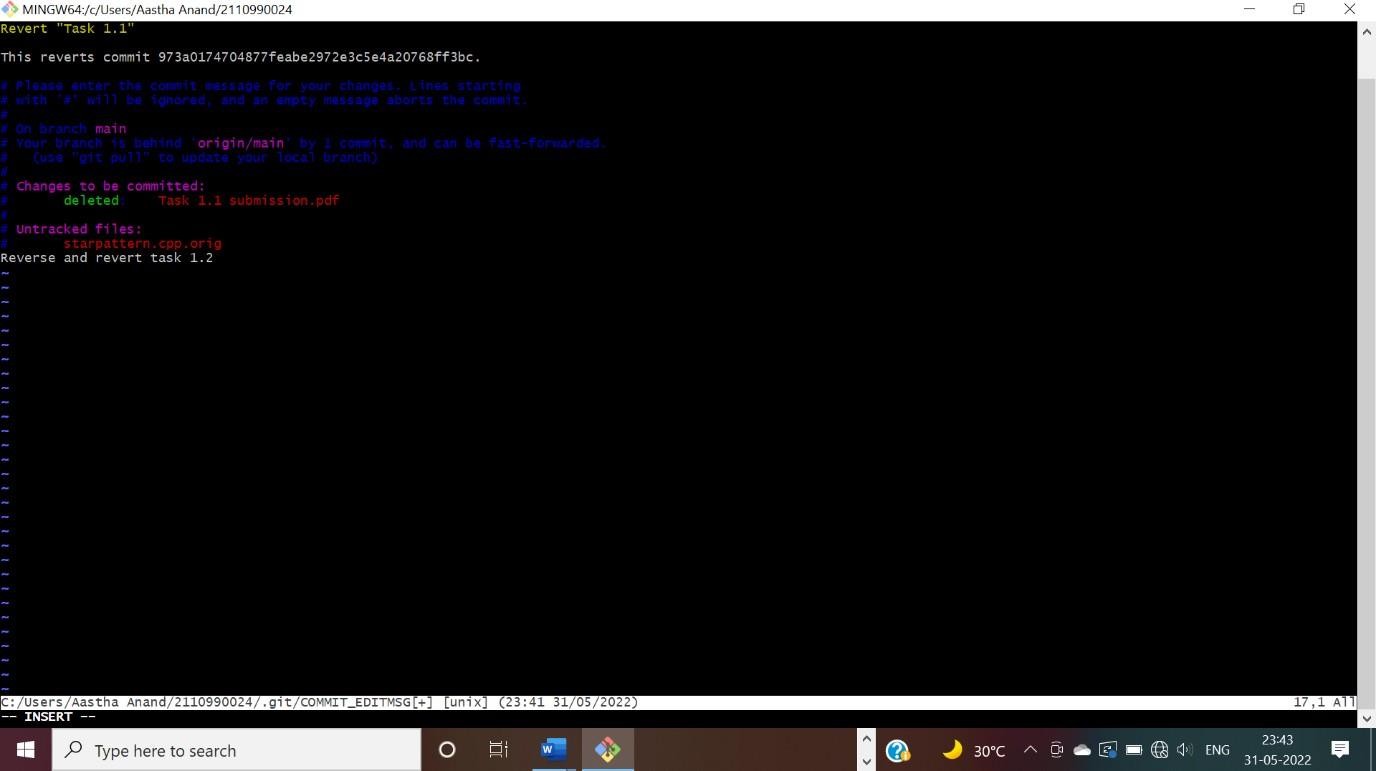
# Revert:

Follow these steps to revert any change:

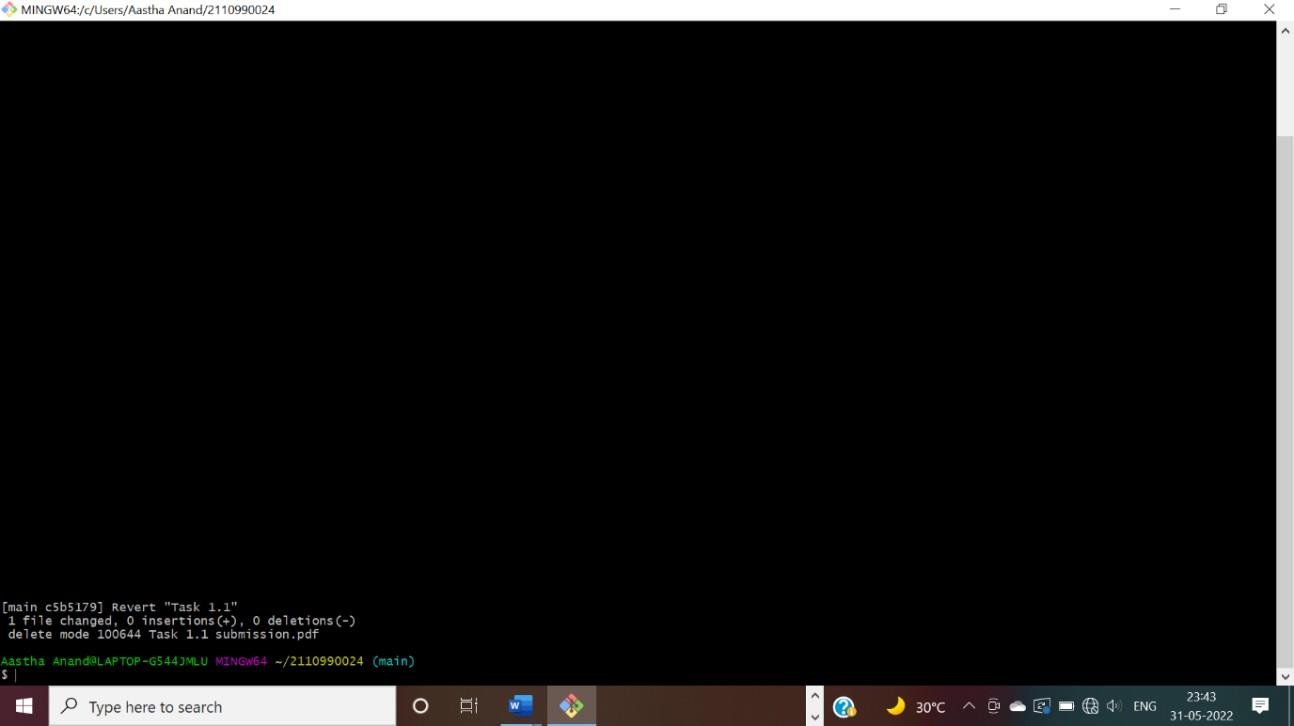
* 1. Pick the change where you want the project to revert back. Copy its checksum and paste it in the revert command.



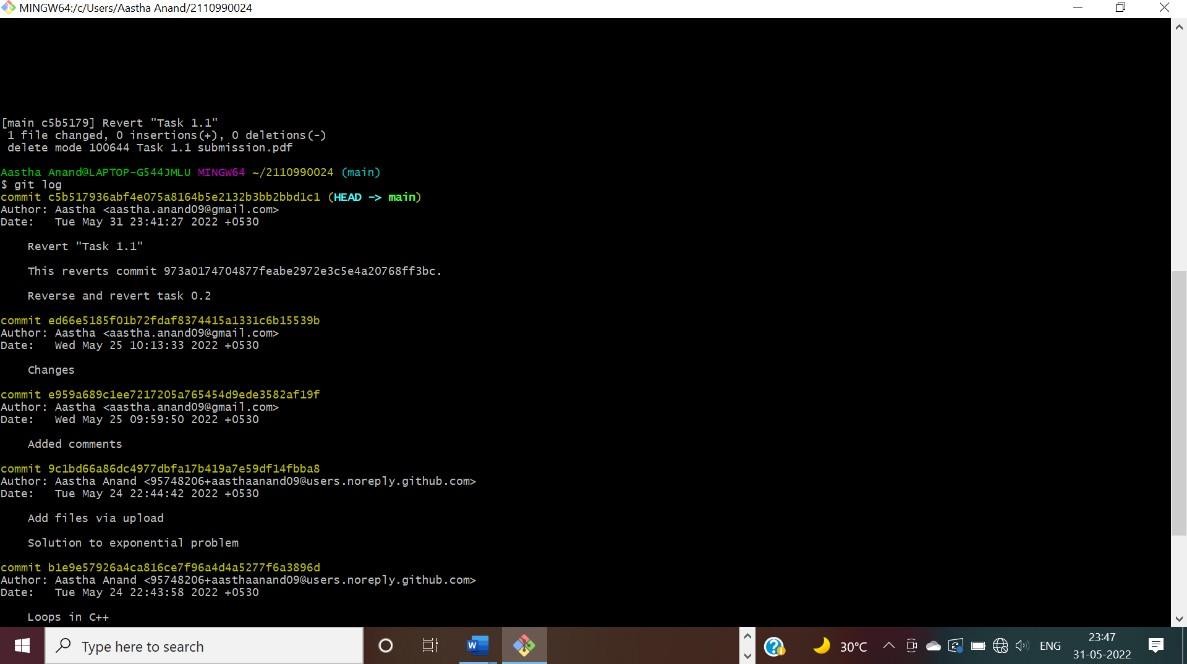
* 1. A window will appear. Press ‘I” and write the statement you want to be displayed for reverting the change.



* 1. After completing press ‘esc’ and write: wq in the terminal.



* 1. Check the git log and you will find another commit is added without affecting the rest commits.



* 1. The change associated to the reverted commit has disappeared.