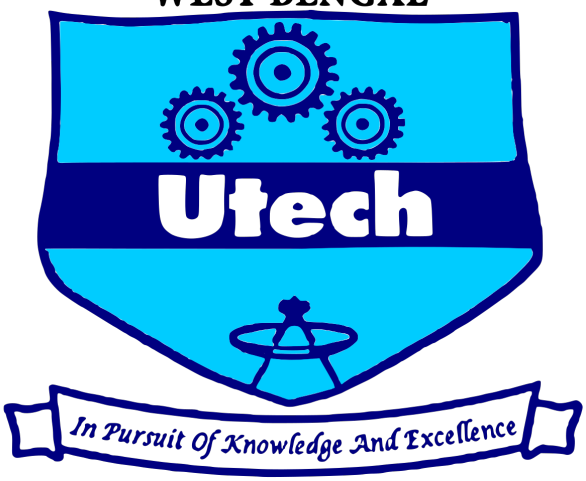
Lab Notebook

Semester: II
Subject: Software Tools and Technology-I Lab
Group Number: 16

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL



Leader Name: Nihar Debnath
Roll Number: 30001223008
Department: B.C.A
GitHub Repo Link: Click me

Collaborators' Names	Roll Numbers	Department
Abir Mandal	30001223066	B.C.A
Anuksha Sarkar	30084323018	BSc in Data Science
Ishita Das	30084323011	BSc in Data Science
Sananda Choudhury	30001223004	B.C.A

Assignment Details

- **Assignment-1:** Create a Git Repository Containing Lab Notebook in a LaTeX File
- **Assignment-2:** Introduction to Latex
- **Assignment-3:** Introduction to Git & Github
- **Assignment-4:** Calculator Program using C
- **Assignment-5:** Converting a submit button to "chin tapak dum dum"
- **Assignment-6:** Demonstrate proficiency in Git branching, merging, and conflict resolution

Contents

Leader: Nihar Debnath

September 17, 2024

1. Introduction to LaTeX

LaTeX is a typesetting system that is widely used for producing scientific and technical documents. It is especially popular for creating documents with complex mathematical formulas, tables, and figures. Unlike word processors, LaTeX provides greater control over document structure and presentation, making it a preferred tool in academia and industry.

2. Key Features of LaTeX

Some of the key features of LaTeX include:

- High-quality typesetting, especially for mathematical and technical content.
- Separation of content and style, allowing users to focus on writing.
- Automated table of contents, bibliography, and cross-references.
- Supports a wide range of document types, including articles, books, reports, and presentations.
- Extensible through packages for additional functionality, such as graphics, tables, and advanced formatting.

3. Why Use LaTeX?

There are several reasons to use LaTeX:

- **Precision**: LaTeX provides exceptional control over document structure and formatting, ensuring that everything looks just right.
- **Mathematical Formulas**: It is the standard for creating documents that contain mathematical symbols and equations.
- **Professional Quality**: Documents created with LaTeX have a professional appearance, suitable for academic papers, theses, and books.
- **Consistency**: Once a style is defined, it is applied consistently throughout the document.
- **Collaboration**: Multiple users can work on the same document without formatting issues, as LaTeX is plain text-based.

4. Creating a LaTeX Repository on GitHub using GitHub Desktop

Version control is crucial for managing changes in documents, especially when working on complex projects. GitHub provides a platform for managing repositories, and GitHub Desktop simplifies working with GitHub.

To create a LaTeX repository on GitHub using GitHub Desktop:

1. **Install GitHub Desktop**: Download and install GitHub Desktop from `https://desktop.github.com/`.
2. **Create a GitHub Account**: If you don't have one, sign up for GitHub at `https://github.com/`.
3. **Create a New Repository**: Open GitHub Desktop and click on *File* → *New Repository*. Name your repository, select the local path, and ensure the repository is initialized with a README.
4. **Clone the Repository**: After creating the repository on GitHub, use GitHub Desktop to clone the repository to your local machine by clicking on *File* → *Clone Repository*.
5. **Add Your LaTeX Files**: Open the cloned folder on your computer and add your LaTeX source files (.tex, .bib, etc.).
6. **Commit and Push**: After adding your files, return to GitHub Desktop, write a commit message, and click on *Commit to main*. Then click on *Push origin* to upload your changes to GitHub.

Figure 1: LaTeX

Member 1: Ishita Das

September 17, 2024

-:GitHub:-

GitHub is a powerful platform for version control and collaborative software development, built on Git, a distributed version control system. It provides a comprehensive environment for managing code repositories, allowing developers to track changes, collaborate on projects, and maintain different versions of their code. Through features like branches and pull requests, GitHub facilitates seamless teamwork by enabling parallel development and peer review of changes before they are integrated into the main project. Additionally, GitHub's issue tracking system helps manage and prioritize tasks, while GitHub Actions offers automation for continuous integration and delivery. With its user-friendly interface and robust toolset, GitHub has become an essential resource for developers, fostering efficient workflows and innovation in the software development community.

:Installation:

Installing GitHub Desktop is a straightforward process that begins with downloading the application from the official GitHub Desktop website. For Windows users, after downloading the .exe installer, you simply run it and follow the on-screen instructions to complete the installation. The process includes selecting installation options and agreeing to the terms of use. For macOS users, you download the .dmg file, open it, and drag the GitHub Desktop icon into the Applications folder to install. Once installed, you can launch the application, where you'll be guided through initial setup steps such as signing in with your GitHub account and configuring basic Git settings. This setup enables you to efficiently manage your repositories and collaborate on projects using the intuitive graphical interface provided by GitHub Desktop.

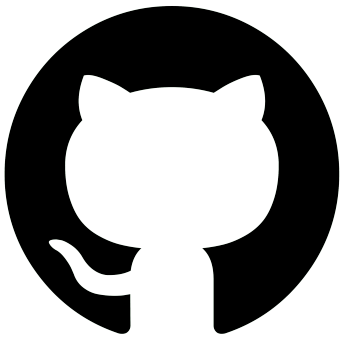


Figure 2: GITHUB

Member 2: Anuksha Sarkar

September 17, 2024

-:What is the purpose of the calculator program:-

The primary purpose of a calculator program is to perform mathematical calculations efficiently and accurately.** It provides a user-friendly interface for entering numbers and operators, and then quickly calculates and displays the result.

Calculators can range from simple devices that handle basic arithmetic to more advanced ones capable of complex functions like trigonometry, statistics, and even programming. They are widely used in various fields, including education, science, engineering, and everyday life.

Code Link: Calculator Program

:Explain the code properly:

Purpose: This C program is designed to function as a simple calculator, allowing users to perform basic arithmetic operations (addition, subtraction, multiplication, and division).

Key Components and Their Functions:

1. Header Inclusion: - *#include < stdio.h >*: This includes the standard input/output library, which provides functions like printf (for printing output) and scanf (for reading input).

2. Main Function: - int main(): This is the entry point of the program. It's where the execution begins.

3. Variable Declarations: - char operator: Stores the operator entered by the user (+, -, *, /). - double num1, num2: Stores the two numbers entered by the user. - double result: Stores the calculated result.

4. User Input: - printf("Enter an operator (+, -, *, /): ");: Prompts the user to enter an operator. - scanf("- Similar prompts and reads are used for the two numbers.

5. Operator Checking and Calculation: - switch (operator): This is a conditional statement that checks the value of the operator variable. - case '+':, case '-':, case '*':, case '/':: These cases handle different operators. - The respective arithmetic operations are performed based on the operator and the results are stored in the result variable. - A check for division by zero is included to prevent errors.

6. Output: - 'printf("Result: ")

Flow of Execution:

1. The program starts execution from the main function. 2. The user is prompted to enter an operator. 3. The entered operator is stored. 4. The user is prompted to enter two numbers. 5. The entered numbers are stored. 6. The switch statement checks the operator and performs the corresponding calculation. 7. The result is printed. 8. The program ends.
Overall, this code provides a basic calculator functionality by taking user input for an operator and two numbers, performing the calculation based on the operator, and displaying the result.

:How to use it, what will be the output after program compilation, set an example:

How to Use:

1. Compile the code: - Save the code in a .c file (e.g., calculator.c). - Open a terminal or command prompt. - Navigate to the directory where you saved the file. - Compile the code using a C compiler like GCC: gcc calculator.c -o calculator This will create an executable file named calculator.

2. Run the program: - Execute the executable file: ./calculator
Example: Enter an operator (+, -, *, /): + Enter two numbers: 5 3 Result: 8.00

In this example: - The user enters + as the operator. - The user enters 5 and 3 as the numbers. - The program calculates 5 + 3 and prints the result 8.00.

Other examples: - - as the operator: 5 - 3 will result in 2.00. - * as the operator: 5 * 3 will result in 15.00. - / as the operator: 5 / 3 will result in 1.67.

Member 3: Sananda Choudhury

September 17, 2024

-:Converting a submit button to "chin tapak dum dum":-

Steps to Fix the Button and Create a Pull Request

1. *Clone the Repository:* - You've already cloned the repository using GitHub Desktop, so you should have the project on your local machine.

2. *Open the Project:* - Open the project in your preferred IDE as per the instructions in the readme.md.

3. *Locate the Button Code:* - Search for the code where the submit button is defined. This is typically found in the frontend code of the application. Depending on the technology stack used (e.g., HTML/CSS, React, Angular, etc.), it could be in a file like index.html, App.js, ButtonComponent.js, or a similar file.

4. *Rename the Button:* - You renamed the button to "Chin Tapak Dum Dum". If the button's size became disproportionate, it's likely due to styling issues.

5. *Test the Application:* - Run the application again to ensure that the button appears correctly and that there are no additional issues.

6. *Commit Your Changes:* - Once the button looks good, commit your changes. Use descriptive commit messages, for example: Fixed button styling after renaming.
bash git add . git commit -m "Fixed button styling after renaming to 'Chin Tapak Dum Dum'"

7. *Push Your Changes:* - Push your changes to your forked repository on GitHub.
bash git push origin main
(Replace main with the correct branch name if it's different.)

8. *Create a Pull Request:* - Go to the original repository on GitHub (the one you cloned from). - You should see an option to create a pull request from your forked repository. Follow the instructions to create a pull request with a title and description of what you have done.
Make sure to mention in the pull request that you have fixed the button styling after renaming it.

Member 4: Abir Mandal

September 17, 2024

Demonstrate proficiency in Git branching, merging, and conflict resolution.

1. Introduction:-
This document outlines the step-by-step process for Git branching, merging, and conflict resolution. The task was completed using the Git version control system, and the repository is hosted on GitHub.
2. Task Steps:-

1. Create a new repository on GitHub called `git-advanced`.
2. Clone the repository to your local machine:

```
git clone https://github.com/your-username/git-advanced.git
cd git-advanced
```

3. Create a new branch called `feature-1` and switch to it:

```
git checkout -b feature-1
```

4. Create a new file called `shared.txt` with the following content:

```
This is a shared file.
Line 1: Original text.
Line 2: Original text.
```

5. Stage and commit the file:

```
git add shared.txt
git commit -m "Add shared.txt with original content"
```

6. Push the branch to GitHub:

```
git push -u origin feature-1
```

7. Create a new branch called `feature-2` and switch to it:

```
git checkout -b feature-2
```

8. Checkout the `shared.txt` file from the main branch:

```
git checkout main -- shared.txt
```

9. Modify the file by changing the second line to:

```
Line 2: Modified text in feature-2.
```

10. Stage and commit the changes:

```
git add shared.txt
git commit -m "Modify Line 2 in shared.txt on feature-2"
```

11. Push the branch to GitHub:

```
git push -u origin feature-2
```

12. Switch back to `feature-1`:

```
git checkout feature-1
```

13. Modify the `shared.txt` file by changing the second line to:

```
Line 2: Modified text in feature-1.
```

14. Stage and commit the changes:

```
git add shared.txt
git commit -m "Modify Line 2 in shared.txt on feature-1"
```

15. Push the branch to GitHub:

```
git push origin feature-1
```

16. Merge `feature-1` into the main branch:

```
git checkout main
git merge feature-1
```

17. Merge `feature-2` into the main branch (this will introduce a conflict):

```
git merge feature-2
```

18. Resolve the conflict in `shared.txt` by editing the file:

```
<<<<<<< HEAD
Line 2: Modified text in feature-1.
=======
Line 2: Modified text in feature-2.
>>>>>>> feature-2
```

After resolving the conflict, it may look like this:

```
Line 2: Modified text from both features.
```

Stage and commit the resolved file:

```
git add shared.txt
git commit -m "Resolve conflict between feature-1 and feature-2"
```

19. Push the updated main branch:

```
git push origin main
```

20. Delete the `feature-1` and `feature-2` branches:

```
git branch -d feature-1
git branch -d feature-2
git push origin --delete feature-1
git push origin --delete feature-2
```

3. Screenshots:-

3.1 GitHub Commit History and Branching:-

We have to include a screenshot of the GitHub repository showing the commit history and branching.

3.2 Git Log and Conflict Resolution:-

We have to include a screenshot of the local machine showing the Git log and the conflict resolution.

4. Write-up of Experience:-

Working with Git branching and merging was insightful. Handling multiple branches like `feature-1` and `feature-2` allowed me to work on different features in parallel. The merge conflict introduced in `shared.txt` was resolved by combining the changes from both branches. Overall, Git's functionality made the process of managing and resolving conflicts straightforward, though it requires careful attention to changes in files.

Clear and meaningful commit messages are crucial to track progress and understand changes in the project.