Software Tools And Technology

Group 2

:

Lab Notebook

Group members:

1. **Rajkumar Pal BCA (Leader)**
2. **Jahir Mondal Bsc in IT(AI)**
3. **Arnab Pandit Bsc in IT(CS)**
4. **Pritam Pratihar BCA**
5. **Suhana Pervaze BCA**

Instructor: Dr.Ayan Ghosh
Course: Software Tools And Technology

Lab Notebook Entries

1 Lab Entry by Rajkumar Pal

1.1 Experiment

Sl. No.	Assignments
1.	Introduction to Github and Github desktop version installation

2 Lab Entry by Jahir Mondal

2.1 Experiment

Sl. No.	Assignments
1.	Converting Submit button to Chin Tapak Dum Dum

3 Lab Entry by Arnab Pandit

3.1 Experiment

Sl. No.	Assignments
1.	Making calculator in C

4 Lab Entry by Pritam Pratihar

4.1 Experiment

Sl. No.	Assignments
1.	Creating latex repository on github

5 Lab Entry by Suhana Pervaze

5.1 Experiment

Sl. No.	Assignments
1.	Introduction to latex

Lab Entry By

Rajkumar Pal

1. Introduction to GitHub and GitHub Desktop Version Installation

In this section, we will explore what GitHub is, its features, and how to install the GitHub Desktop version. GitHub is a platform that enables developers to collaborate, track, and manage code versions through Git version control. It provides a web-based interface for easier management of Git repositories.

1. Introduction to GitHub

GitHub is a web-based platform that allows developers to host and review code, manage projects, and build software collaboratively. Git is a version control system that enables tracking of changes, and GitHub adds a graphical interface and additional features on top of Git.

2. GitHub Desktop Installation

GitHub Desktop is a GUI client that simplifies the interaction with Git and GitHub. It allows users to handle repositories,

create branches, and manage commits without using the command line.

Steps to Install GitHub Desktop:

1. **Visit the GitHub Desktop official website at** `https://desktop.github.com/`**.**
2. **Click on the** `Download for your OS` **button (Windows/Mac).**
3. **Follow the installation instructions provided by the installer.**
4. **Once installed, sign in with your GitHub credentials to start using GitHub Desktop.**

Lab Entry By

Jahir Mondal

2. Changing the submit button to Chin Tapak Dum Dum and fixing the disproportionate

6 STEPS

Changed the submit button to"Chin tapak Dum Dum through this code.

```
add(symbolPanel, BorderLayout.CENTER);

// Panel for submit button
Panel controlPanel = new Panel(new FlowLayout());
submitButton = new Button("Chin Tapak Dum Dum");
submitButton.setFont(new Font("Arial", Font.BOLD, 20));
submitButton.setBackground(Color.RED);
submitButton.setForeground(Color.WHITE);
submitButton.addActionListener(this);
controlPanel.add(submitButton);
add(controlPanel, BorderLayout.SOUTH);
```

Figure 1: JAVA CODE

- **Font Size and Style: The font size and style have been adjusted for improved readability and consistency with the overall design.**
- **Background Color: The background color of the button has been updated to create a more visually appealing and cohesive look.**
- **Font Color: The font color has been modified to ensure strong contrast with the background, enhancing legibility.**
- **Element Proportions: Any disproportionate elements have been corrected to achieve a more balanced and aesthetically pleasing design.**

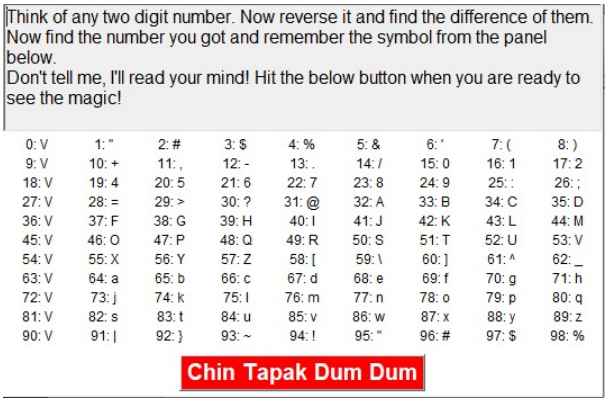

Figure 2: FINAL OUTPUT

7 OUTPUT

3. Calculator in C

Lab Entry By

Arnab Pandit

8 Introduction

In this document, I will present a detailed explanation of a simple calculator program developed in the C programming language. This calculator performs various arithmetic operations including addition, subtraction, multiplication, division, percentage calculation, squaring, and cubing of numbers. The design of this calculator is aimed at providing a user-friendly interface with robust input validation to ensure accurate results.

9 Code

Below is the complete code for the calculator in C, followed by an explanation of each function and its role in the program.

Listing 1: Simple Calculator in C

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

// Function declarations
void addition();
void subtract();
void multiply();
void divide();
void percentage();
void square();
void cube();

int main() {
    int op;

    do {
        printf("Select an operation to perform in the C Calculator
            :\n");
        printf("1. Addition\n");
        printf("2. Subtraction\n");
        printf("3. Multiplication\n");
        printf("4. Division\n");
        printf("5. Percentage\n");
        printf("6. Square\n");
        printf("7. Cube\n");
        printf("8. Exit\n");
        printf("Please make a choice: ");

        // Validate user input
        while (scanf("%d", &op) != 1) {
```

```c
            printf("Invalid input! Please enter a number between 1 and 8: ");
            while (getchar() != '\n'); // Clear the invalid input
        }

        // Perform the selected operation
        switch (op) {
            case 1:
                addition(); // Call the addition function
                break;
            case 2:
                subtract(); // Call the subtraction function
                break;
            case 3:
                multiply(); // Call the multiplication function
                break;
            case 4:
                divide(); // Call the division function
                break;
            case 5:
                percentage(); // Call the percentage function
                break;
            case 6:
                square(); // Call the square function
                break;
            case 7:
                cube(); // Call the cube function
                break;
            case 8:
                printf("Exiting the program.\n");
                exit(0); // Exit the program
            default:
                printf("Error! Invalid choice. Try again.\n");
        }
        printf("\n***************\n");
    } while (op != 8); // Repeat until the user chooses to exit

    return 0;
}

// Function definitions

// Function to add numbers
void addition() {
    int i, num;
    double sum = 0;
    printf("How many numbers do you want to add? ");

    // Read the number of inputs and validate it
    while (scanf("%d", &num) != 1 || num <= 0) {
        printf("Invalid input! Enter a positive number: ");
        while (getchar() != '\n'); // Clear the invalid input
    }

    printf("Enter the numbers:\n");
    for (i = 1; i <= num; i++) {
        double f_num;
        while (scanf("%lf", &f_num) != 1) {
```

```
            printf("Invalid input! Enter a valid number: ");
            while (getchar() != '\n'); // Clear the invalid input
        }
        sum += f_num;
    }

    // Display the result
    printf("Total sum of the numbers = %.2lf\n", sum);
}

// Function to subtract two numbers
void subtract() {
    double n1, n2;
    printf("Enter the first number: ");
    scanf("%lf", &n1);
    printf("Enter the second number: ");
    scanf("%lf", &n2);
    printf("The result of %.2lf - %.2lf is: %.2lf\n", n1, n2, n1 -
        n2);
}

// Function to multiply two numbers
void multiply() {
    double n1, n2;
    printf("Enter the first number: ");
    scanf("%lf", &n1);
    printf("Enter the second number: ");
    scanf("%lf", &n2);
    printf("The result of %.2lf * %.2lf is: %.2lf\n", n1, n2, n1 *
        n2);
}

// Function to divide two numbers
void divide() {
    double n1, n2;
    printf("Enter the first number: ");
    scanf("%lf", &n1);
    printf("Enter the second number: ");
    scanf("%lf", &n2);

    if (n2 != 0) {
        printf("The result of %.2lf / %.2lf is: %.2lf\n", n1, n2,
            n1 / n2);
    } else {
        printf("Error! Division by zero is not allowed.\n");
    }
}

// Function to calculate the percentage
void percentage() {
    double value, percent;
    printf("Enter the value: ");
    scanf("%lf", &value);
    printf("Enter the percentage: ");
    scanf("%lf", &percent);
    printf("%.2lf percent of %.2lf is: %.2lf\n", percent, value, (
        percent / 100) * value);
}
```

```c

// Function to calculate the square of a number
void square() {
    double n1;
    printf("Enter a number to get the square: ");
    scanf("%lf", &n1);
    printf("The square of %.2lf is: %.2lf\n", n1, n1 * n1);
}

// Function to calculate the cube of a number
void cube() {
    double n1;
    printf("Enter a number to get the cube: ");
    scanf("%lf", &n1);
    printf("The cube of %.2lf is: %.2lf\n", n1, n1 * n1 * n1);
}
```

10 Function Explanations With Output Values

10.1 Addition Function

The `addition()` function allows the user to input multiple numbers and calculates their sum. It prompts the user for the number of inputs, validates the input, and then reads the numbers, summing them up.

Example Output:

```
Select an operation to perform in the C Calculator:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Percentage
6. Square
7. Cube
8. Exit
Please make a choice: 1
How many numbers do you want to add? 3
Enter the numbers:
5
10
15
Total sum of the numbers = 30.00
```

10.2 Subtraction Function

The `subtract()` function performs subtraction between two user-provided numbers. It prompts the user for two numbers and then calculates and displays their difference.

Example Output:

```
Select an operation to perform in the C Calculator:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Percentage
6. Square
7. Cube
8. Exit
Please make a choice: 2
Enter the first number: 20
Enter the second number: 5
The result of 20.00 - 5.00 is: 15.00
```

10.3 Multiplication Function

The `multiply()` function performs multiplication of two numbers entered by the user. It displays the product of the two numbers.

Example Output:

```
Select an operation to perform in the C Calculator:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Percentage
6. Square
7. Cube
8. Exit
Please make a choice: 3
Enter the first number: 4
Enter the second number: 5
The result of 4.00 * 5.00 is: 20.00
```

10.4 Division Function

The `divide()` function handles the division of two numbers. It includes a check to prevent division by zero, ensuring the user does not encounter an error.

Example Output:

```
Select an operation to perform in the C Calculator:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Percentage
6. Square
7. Cube
8. Exit
Please make a choice: 4
```

```
Enter the first number: 25
Enter the second number: 5
The result of 25.00 / 5.00 is: 5.00
```

10.5 Percentage Function

The `percentage()` function calculates the percentage of a given value. The user inputs a value and the percentage, and the function computes and displays the result.

Example Output:

```
Select an operation to perform in the C Calculator:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Percentage
6. Square
7. Cube
8. Exit
Please make a choice: 5
Enter the value: 200
Enter the percentage: 15
15.00 percent of 200.00 is: 30.00
```

10.6 Square Function

The `square()` function calculates the square of a number. The user inputs a number, and the function computes and displays its square.

Example Output:

```
Select an operation to perform in the C Calculator:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Percentage
6. Square
7. Cube
8. Exit
Please make a choice: 6
Enter a number to get the square: 7
The square of 7.00 is: 49.00
```

10.7 Cube Function

The `cube()` function calculates the cube of a number. The user inputs a number, and the function computes and displays its cube.

Example Output:

```
Select an operation to perform in the C Calculator:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Percentage
6. Square
7. Cube
8. Exit
Please make a choice: 7
Enter a number to get the cube: 3
The cube of 3.00 is: 27.00
```

11 Conclusion

In conclusion, the simple calculator program developed in C provides a comprehensive tool for performing basic arithmetic operations. This calculator covers a range of functions including addition, subtraction, multiplication, division, percentage calculation, and the computation of squares and cubes of numbers. Each function is designed with user-friendly prompts and robust input validation to ensure accuracy and ease of use.
The program demonstrates effective use of functions to modulate the code, making it both organized and easy to maintain. By implementing input validation and error handling, the calculator minimizes the risk of user errors and enhances the overall user experience.
This project not only highlights fundamental programming concepts such as function definition and control structures but also showcases practical application in developing tools for everyday use. This simple calculator serves as a solid foundation for building more complex applications and improving programming skills in C.
Overall, this calculator is a valuable exercise in programming, offering insights into both basic and intermediate concepts, and providing a useful utility for performing arithmetic operations efficiently.

4. Introduction to Latex

Lab Entry By

Pritam Pratihar

12 Introduction

LaTeX is a sophisticated typesetting system developed by Leslie Lamport and built upon the TeX typesetting system created by Donald Knuth. It is widely recognized for its capability to handle complex formatting tasks with precision, making it the preferred choice for producing high-quality scientific, mathematical, and academic documents.
LaTeX excels in the following areas:

- **Mathematical Typesetting: LaTeX is renowned for its exceptional ability to render complex mathematical equations and symbols. It provides a wide range of mathematical symbols and structures, such as fractions, integrals, summations, and matrices, all formatted with high typographical quality. This capability makes LaTeX an essential tool for researchers and academics who need to include precise and well-formatted mathematical content in their documents.**
- **Bibliographies and Citations: Managing bibliographies and citations is another area where LaTeX excels. Through the use of packages like bibtex or biber, users can automate the generation of bibliographies and manage references efficiently. This feature is particularly useful for academic writing, where proper citation and referencing are crucial.**
- **Document Structuring: LaTeX allows users to structure their documents in a hierarchical manner, using sections, subsections, and subsubsections. This hierarchical structure ensures that documents are well-organized and easy to navigate. Users can also create tables of contents, lists of figures, and lists of tables automatically based on the document's structure.**
- **Customization and Flexibility: With LaTeX, users have extensive control over document formatting and layout. Customizing fonts, margins, line spacing, and other typographical elements is straightforward. This level of control allows for the creation of highly customized and professional-looking documents tailored to specific requirements.**
- **Cross-Referencing and Indexing: LaTeX provides robust features for cross-referencing sections, figures, tables, and equations. It also supports automated indexing, making it easier to create comprehensive indexes and reference materials for larger documents.**

LaTeX is not limited to scientific and academic papers; it is also used for creating a variety of other document types, including:

- **Technical Reports: Detailed and well-organized reports on technical subjects, often including complex data and visualizations.**

- **Theses and Dissertations: Formal academic documents that require precise formatting, including chapters, appendices, and bibliographies.**
- **Books and Articles: Publications that benefit from LaTeX's ability to handle large documents and complex layouts.**
- **Presentations: Using the beamer package, LaTeX can create professional and visually appealing presentation slides.**

Despite its powerful features, LaTeX can have a steep learning curve, especially for those new to typesetting or programming. However, once mastered, it provides a level of control and flexibility that is unmatched by traditional word processors. Many users find that investing time in learning LaTeX pays off in the form of high-quality, professional documents that meet rigorous standards.
In summary, LaTeX is a versatile and powerful typesetting system that supports a wide range of document types and formatting requirements, making it an invaluable tool for academics, scientists, and professionals who need to produce high-quality, structured documents.

13 Basic Document Structure

A basic LaTeX document has a fundamental structure that consists of several key components. Understanding this structure is essential for creating and managing LaTeX documents effectively. Below is an elaboration on each part of this structure:

```
\documentclass{article}
```

Document Class: The `\documentclass{article}` command specifies the type of document you are creating. The article class is one of the most commonly used document classes, suitable for shorter documents such as journal articles, reports, and essays. Other document classes include report, which is ideal for longer documents with chapters, and book, which is designed for books and includes additional features for managing chapters and sections.

```
\begin{document}
```

Document Environment: The `\begin{document}` and `\end{document}` commands define the main content area of your document. All of your text, sections, figures, tables, and other content must be placed between these two commands. Anything outside of this environment is not processed as part of the document's content.

```
% Your content here
```

Content: Within the document environment, you can include various elements to build your document. Here are some common elements you might use:

- **Sections and Subsections: Use commands like `\section{Title}`, `\subsection{Subtitle}`, and `\subsubsection{Subsubtitle}` to organize your document into a hierarchical structure.**
- **Text and Paragraphs: Simply type your text as you would in any other word processor. LaTeX will automatically format it into paragraphs.**

- **Figures and Tables: You can include figures and tables using the figure and table environments, respectively, and manage their placement and captions.**
- **Mathematics: Insert mathematical expressions and equations using `\begin{equation}` for block equations or inline math using ....**
- **References and Citations: Use bibliographic tools to manage references and citations in your document.**

Additional Packages: LaTeX allows for the inclusion of additional packages that extend its functionality. For example, to include graphics, you might add `\usepackage{graphicx}` in the preamble (before `\begin{document}`). Packages provide extra features and commands that are not included in the base LaTeX installation.
In summary, the basic structure of a LaTeX document involves defining the document class, including content within the document environment, and optionally utilizing additional packages to enhance functionality. Understanding and utilizing these components allows you to create well-structured and professionally formatted documents.

14 Text Formatting

LaTeX provides a range of commands to format text according to your needs. Understanding these commands will help you produce well-structured and visually appealing documents. Below are some commonly used text formatting commands with elaborations:

- **Bold Text:**

  ```
  \textbf{This is bold text}
  ```

  To create bold text in LaTeX, use the `\textbf{}` command. This command is often used to emphasize important words or sections of text. The text enclosed within the braces will be rendered in bold typeface, which helps to draw attention to key parts of your document.

- ***Italic Text:***

  ```
  \textit{This is italic text}
  ```

  Italic text is created using the `\textit{}` command. This formatting style is typically used for emphasis, highlighting, or denoting terms that are being defined or used in a special way. Text enclosed in `\textit{}` will appear in an italic typeface, making it distinct from the surrounding text.

- ***Underlined Text:***

  ```
  \underline{This is underlined text}
  ```

To underline text, use the \underline{} command. Underlining is less common in modern documents compared to bold or italic text, but it can be useful for emphasizing specific parts of text or denoting URLs. The text inside the braces will be displayed with a horizontal line beneath it.

- Typewriter or Monospaced Font:

```
\texttt{This is monospaced text}
```

The \texttt{} command formats text in a typewriter or monospaced font. This font style is often used for code snippets or computer output, where each character takes up the same amount of space horizontally. It provides a clear and consistent appearance for text that needs to align precisely.

- **Sans-Serif Font:**

```
\textsf{This is sans-serif text}
```

The \textsf{} command changes the font to sans-serif. Sans-serif fonts are characterized by the absence of the small projecting features (serifs) at the ends of strokes. They are commonly used for modern, clean-looking documents and are often preferred for headings and titles.

- Small Caps:

```
\textsc{This is small caps text}
```

The \textsc{} command formats text in small capitals. In small caps, lowercase letters are rendered in uppercase but with a smaller size. This style is used for emphasis or to distinguish text from the surrounding content.

- **Text in Different Colors:**

```
\usepackage{xcolor}
\textcolor{red}{This is red text}
```

To color text, use the \textcolor{color}{text} command. First, ensure that you include the xcolor package in your document preamble with \usepackage{xcolor}. You can then specify the desired color (e.g., red, blue, green) and the text to be colored.

- Custom Font Sizes:

```
{\small This text is smaller.}
{\large This text is larger.}
```

Font sizes can be adjusted using commands like , , and other size commands. The text enclosed within these commands will be displayed in the specified size relative to the default size.

By mastering these text formatting commands, you can effectively highlight important information, create a visually appealing document, and ensure that your content is presented in a clear and organized manner.

15 Mathematical Equations

One of the most powerful features of LaTeX is its ability to typeset complex mathematical equations with precision and clarity. LaTeX provides a range of tools for displaying mathematical formulas and expressions, making it a popular choice for academic and scientific documents.

15.1 Displayed Equations

Displayed equations are centered on their own line and are typically used for important or complex equations that need to stand out. In LaTeX, you can create displayed equations using the `equation` environment, which automatically numbers the equations for reference in the document. For example:

$$E = mc^2 \tag{1}$$

This example demonstrates Einstein's famous mass-energy equivalence formula. The equation is centered and displayed on its own line, and LaTeX automatically adds a number to the right of the equation for easy reference.
To write more complex equations or include multiple equations in a sequence, you can use other environments such as `align` or `gather`. For example:

$$a^2 + b^2 = c^2 \tag{2}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3}$$

The `align` environment allows you to align equations at the equal signs or other operators, while the `gather` environment centers multiple equations without alignment.

15.2 Inline Equations

Inline equations are used within a line of text and are typically employed for simpler expressions or mathematical notations. To include an inline equation, you use the `$` symbols around the mathematical expression. For example:

```
The Pythagorean theorem states that $a^2 + b^2 = c^2$.
```

This renders as: The Pythagorean theorem states that $a^2 + b^2 = c^2$.
Inline equations are useful for incorporating mathematical expressions into sentences without disrupting the flow of text. For more complex expressions, you may still need to use displayed equations to ensure clarity.

15.3 Mathematical Symbols and Operators

LaTeX provides a rich set of symbols and operators for mathematical notation. Here are a few examples:

- **Greek letters:** α, β, γ **(use commands like `\alpha`, `\beta`)**
- **Operators:** $\sum, \int, \prod$ **(use commands like `\sum`, `\int`)**
- **Relations:** $\leq, \geq, \neq$ **(use commands like `\leq`, `\geq`)**

- **Special symbols: ∞, ∇, ∂ (use commands like `\infinity`, `\nabla`, `\partial`)**

These symbols and operators are essential for expressing mathematical concepts and equations clearly and accurately in LaTeX.

15.4 Using Packages for Advanced Math

For more advanced mathematical typesetting, LaTeX offers several packages such as amsmath and mathtools. These packages provide additional functionality and formatting options. For instance, to use the amsmath package, include the following line in the preamble of your document:

```
\usepackage{amsmath}
```

The amsmath package adds features like `cases` for piecewise functions, `align*` for unnumbered equations, and many more.
In summary, LaTeX excels at handling mathematical notation with its comprehensive set of tools and features. Whether you're working with simple inline equations or complex displayed formulas, LaTeX provides the flexibility and precision needed for professional-quality typesetting.

16 Inserting Images

Inserting images into a LaTeX document enhances the visual appeal and can help illustrate concepts or provide examples. To include images, you need to use the `graphicx` package, which provides the necessary functionality. Here's a step-by-step guide on how to do this:

16.1 Including the Graphicx Package

Before you can insert images, you need to include the `graphicx` package in your document preamble. Add the following line to the preamble of your document:

```
\usepackage{graphicx}
```

This line tells LaTeX to load the package that handles image inclusion.

16.2 Basic Image Insertion

To insert an image, you use the `figure` environment along with the `includegraphics` command. Here's a basic example:

```
\begin{figure}[h]
    \centering
    \includegraphics[width=0.5\textwidth]{example-image}
    \caption{An example image.}
    \label{fig:example}
\end{figure}
```

This code snippet does the following: - `figure`[h]: The `figure` environment is used to include images. The optional argument `[h]` is a placement specifier indicating that LaTeX should try to position the figure "here" (i.e., approximately at the location where it appears in the text). - : This command centers the image horizontally within the figure environment. - : This command inserts the image file named `example-image`. The optional argument `width=0.5` scales the image to 50- `An example image.`: Adds a caption below the image. This caption will be numbered and can be referenced elsewhere in the document. - : Sets a label for the figure, allowing you to reference it in the text using `16.2`.

16.3 Advanced Options

The `graphicx` package provides additional options for controlling image placement and appearance. Here are a few advanced options:

- `scale`: Scale the image by a factor. For example, scales the image to 75
- `height` and `width`: Specify the height or width of the image. For instance, sets the height of the image to 5 cm, while preserving the aspect ratio.
- `angle`: Rotate the image by a specified angle. For example, rotates the image 90 degrees clockwise.

16.4 Figure Placement

By default, LaTeX will try to place figures in a location that optimizes the document layout. The placement specifiers `[h]`, `[t]`, `[b]`, and `[p]` can be used to suggest figure placement:

- `h`: Here, approximately at the point in the text where it is defined.
- `t`: Top of the page.
- `b`: Bottom of the page.
- `p`: On a separate page for floats.

LaTeX may still move figures to optimize the layout, so it is a good practice to place figures in a logical and consistent manner throughout your document.

16.5 Including Images from External Sources

When inserting images, ensure that the image file is in a format supported by LaTeX, such as PNG, JPEG, or PDF. Place the image file in the same directory as your .tex file or provide a relative path to its location.
By following these steps, you can effectively include and manage images in your LaTeX document, enhancing the presentation and clarity of your work.

17 Creating Lists

In LaTeX, you can create both numbered (ordered) and bulleted (unordered) lists to organize content effectively. Lists are useful for presenting information in a structured and readable format. LaTeX provides straightforward commands to create these lists.

17.1 Bulleted List

Bulleted lists are used when the order of items is not important. They are created using the `itemize` environment. Each item in the list is preceded by a bullet point. Here is a basic example:

```
\begin{itemize}
    \item First item
    \item Second item
    \item Third item
\end{itemize}
```

This will render as:

- **First item**
- **Second item**
- **Third item**

You can customize the appearance of the bullet points by using various LaTeX packages or by modifying the list style. For instance, the `enumitem` package allows you to change the symbols or styles used for bullet points.

17.2 Numbered List

Numbered lists are used when the order of items is significant. They are created using the `enumerate` environment. Each item in the list is preceded by a number. Here is a basic example:

```
\begin{enumerate}
    \item First item
    \item Second item
    \item Third item
\end{enumerate}
```

This will render as:

1. **First item**
2. **Second item**
3. **Third item**

The `enumerate` environment automatically numbers the items and adjusts the numbering as you add or remove items. Like bulleted lists, you can also customize numbered lists using the `enumitem` package to change numbering styles, such as using Roman numerals or letters.

17.3 Nested Lists

You can create nested lists to further organize content. This is done by placing one list environment inside another. Here's an example of a nested bulleted list:

```
\begin{itemize}
    \item Main item
    \begin{itemize}
        \item Sub-item 1
        \item Sub-item 2
    \end{itemize}
    \item Another main item
\end{itemize}
```

This will render as:

- **Main item**
  - **Sub-item 1**
  - **Sub-item 2**
- **Another main item**

Similarly, you can nest numbered lists:

```
\begin{enumerate}
    \item Main item
    \begin{enumerate}
        \item Sub-item 1
        \item Sub-item 2
    \end{enumerate}
    \item Another main item
\end{enumerate}
```

This will render as:

1. **Main item**
   (a) **Sub-item 1**
   (b) **Sub-item 2**
2. **Another main item**

Nested lists are helpful for breaking down complex information into more digestible parts, and LaTeX handles the indentation and formatting automatically.

18 Adding Hyperlinks

Adding hyperlinks in a LaTeX document can enhance its usability by allowing readers to navigate to external websites, documents, or specific locations within the same document. This is particularly useful for creating interactive and web-friendly documents.
To add hyperlinks, you need to use the `hyperref` package, which provides a simple interface for creating hyperlinks. Here's a step-by-step explanation of how to use it:

- **First, include the `hyperref` package in the preamble of your document by adding the line:**

  ```
  \usepackage{hyperref}
  ```

  This package enables hyperlink functionality throughout your document.

- **To create an external link, use the `\href` command. The syntax for this command is:**

  ```
  \href{URL}{link text}
  ```

  Where `URL` is the web address you want to link to, and `link text` is the text that will appear as the clickable link.

- **For example, to link to the LaTeX project website, you would use:**

  ```
  \href{https://www.latex-project.org/}{Visit the \LaTeX{} project website.}
  ```

  This command creates a hyperlink with the text "Visit the LaTeX project website" that directs to the specified URL.

- **You can also link to sections within the same document. To do this, use the `\label` command to create a reference point and the `\ref` command to link to it. For instance:**

  ```
  \section{Introduction}\label{sec:intro}


  Refer to Section \ref{sec:intro} for an introduction.
  ```

  This will create a link in the text that points to the "Introduction" section within the document.

- **For internal links to specific locations, such as figures or tables, you can use the `\pageref` command along with `\label`:**

  ```
  \begin{figure}[h]
      \centering
      \includegraphics[width=0.5\textwidth]{example-image}
      \caption{An example image.}
      \label{fig:example}
  \end{figure}


  See Figure \ref{fig:example} on page \pageref{fig:example}.
  ```

This command will link to the figure and provide the page number where it is located.

By incorporating hyperlinks, you can make your LaTeX documents more interactive and user-friendly, providing easy access to additional resources and information. The `hyperref` package also allows for customization of link colors and styles, which can be controlled through package options, enhancing the visual appeal and functionality of your document.

19 Conclusion

This document has provided an introduction to the fundamental features of LaTeX, a powerful typesetting system used for creating high-quality documents. We covered essential topics such as basic document structure, text formatting, mathematical equations, and image insertion, laying the groundwork for more advanced LaTeX capabilities.

LaTeX offers extensive features beyond those discussed, including sophisticated bibliographic management with packages like biblatex or natbib, and advanced table and figure formatting. It also supports custom commands and environments, enhancing document consistency and efficiency.

Additionally, LaTeX accommodates internationalization and localization through packages such as polyglossia or babel, and allows document output in various formats like PDF, HTML, and EPUB.

In summary, while this document provides a basic overview, LaTeX's true strength lies in its advanced features and flexibility, enabling the creation of professional-quality documents with precision. As you delve deeper into LaTeX, you'll uncover more tools and techniques that can enhance your document preparation process.

4. Creating LaTeX Repository on GitHub

Lab Entry By

Suhana Pervaze

20 Introduction

In this guide, we will walk through the steps to create a LaTeX repository on GitHub. GitHub is a popular platform for version control and collaboration, allowing you to store your LaTeX projects and share them with others.

21 Pre-requisites

Before proceeding, make sure you have the following:

- **A GitHub account (https://github.com/).**
- `git` **installed on your machine.**
- **Basic understanding of version control.**

22 Step 1: Creating a Repository

To create a repository on GitHub:

1. **Log in to your GitHub account.**
2. **Click on the New repository button in the top right corner.**
3. **Give your repository a name, e.g.,** `my-latex-project`**.**
4. **Optionally, add a description.**
5. **Choose whether you want the repository to be public or private.**
6. **Click Create repository.**

23 Step 2: Setting Up LaTeX Project Locally

Next, set up your LaTeX project locally:

1. **Navigate to the folder where you want to store your project.**
2. **Initialize a** `git` **repository:**

```
git init
```

3. **Create your LaTeX files, for example:**

```
touch main.tex
```

24 Step 3: Connecting Local Repository to GitHub

Now, connect your local repository to GitHub:

1. **In your terminal, run:**

   ```
   git remote add origin https://github.com/your-username/my-latex-project.git
   ```

2. **Add and commit your changes:**

   ```
   git add .
   git commit -m "Initial commit"
   ```

3. **Push your changes to GitHub:**

   ```
   git push -u origin master
   ```

25 Step 4: Managing Your LaTeX Project

Once the repository is created and set up, you can continue to work on your LaTeX project by adding, committing, and pushing changes. GitHub also allows for collaborative work, so you can invite others to contribute to your project.

25.1 Adding Files

To add new files, such as additional LaTeX chapters or images, you can:

```
git add newfile.tex
git commit -m "Added new chapter"
git push
```

25.2 Collaboration

To collaborate with others, simply add them as collaborators in your GitHub repository settings, or share the repository link for them to fork.

26 Conclusion

In this document, we covered how to create a LaTeX repository on GitHub and manage it using basic `git` commands. By hosting your LaTeX projects on GitHub, you can leverage version control and collaboration features to enhance your workflow.