Lab Notebook

Team 1

Maulana Abul Kalam Azad University of Technology

Software Tools and Techniques - Lab Notebook

Assignment Details

- **Assignment:** Create a Git Repository Containing a Lab Notebook in LaTeX Format
- **Subject:** Software Tools and Techniques
- **Team No.:** 1
- **GitHub Repo Link:** `https://github.com/MazidNawaz/Group_1_Latex.git`

Team Members

- **Member 1 (Lead):**
  - **Name: Mazid Nawaz Ahmad**
  - **Reg No.: 233002410602**
  - **Course: Bsc. IT Data Science**
  - **GitHub Link:** `https://github.com/MazidNawaz`
- **Member 2:**
  - **Name: Ratul Mondal**
  - **Reg No.: 233002410593**
  - **Course: Bsc. It Data Science**
  - **GitHub Link:** `https://github.com/CLOUD77782`
- **Member 3:**

  - **Name: Srizani Dutta**
  - **Reg No.: 233002410005**
  - **Course: Bsc. Forensic Science**
  - **GitHub Link:** `https://github.com/srizani04`

- **Member 4:**
  - **Name: Sneha Ghosh**
  - **Reg No.: 233002410025**
  - **Course: Bsc. Forensic Science**
  - **GitHub Link:** `https://github.com/Sneha-25-ghosh`

- **Member 5:**
  - **Name: Safa Ahmad**
  - **Reg No.: 233002410559**
  - **Course: BSc AI**
  - **GitHub Link:** `https://github.com/safaahmad7`

Table of Contents

Contents

1 Lab 1: Calculator Program using C

1.1 Objective

The objective of this lab is to develop a basic calculator program using the C programming language. The calculator will perform simple arithmetic operations like addition, subtraction, multiplication, and division based on user input.

1.2 Program Overview

The calculator program is designed to:

- Accept two numbers from the user.
- Prompt the user to select an arithmetic operation (Addition, Subtraction, Multiplication, Division).
- Perform the selected operation.
- Display the result of the operation to the user.

The program includes error handling to manage division by zero and other invalid inputs.

1.3 Code Implementation

The following is the C code for the calculator program:

```
#include <stdio.h>

int main() {
    char operator;
    double num1, num2, result;

    printf("Enter an operator (+, -, *, /): ");
    scanf("%c", &operator);

    printf("Enter two operands: ");
    scanf("%lf %lf", &num1, &num2);

    switch(operator) {
        case '+':
            result = num1 + num2;
            break;
        case '-':
            result = num1 - num2;
            break;
        case '*':
            result = num1 * num2;
            break;
        case '/':
```

```
            if (num2 != 0)
                result = num1 / num2;
            else {
                printf("Error! Division by zero.\n");
                return -1;
            }
            break;
        default:
            printf("Error! Operator is not correct\n");
            return -1;
    }

    printf("Result: %.2lf\n", result);
    return 0;
}
```

1.4 Compiling and Running the Program

To compile and run the calculator program:

1. Open a terminal or command prompt.
2. Navigate to the directory where the C file is located.
3. Compile the program using a C compiler (e.g., GCC):

```
gcc calculator.c -o calculator
```

4. Run the compiled program:

```
./calculator
```

1.5 Adding the Calculator Program to GitHub Repository

To add this calculator program to a GitHub repository, follow these steps:

1.5.1 Step 1: Initialize a Local Git Repository

1. Open the terminal and navigate to the directory where your `calculator.c` file is located.
2. If you haven't already, initialize a Git repository in that directory:

```
git init
```

This command creates a new Git repository in the current directory.

1.5.2 Step 2: Add the File to the Repository

1. Add the `calculator.c` file to the staging area:

   ```
   git add calculator.c
   ```

   This command stages the file, indicating that you want to include it in the next commit.

1.5.3 Step 3: Commit the Changes

1. Commit the file to the repository with a meaningful message:

   ```
   git commit -m "Add calculator program in C"
   ```

1.5.4 Step 4: Push the Changes to GitHub

1. Link your local repository to a remote GitHub repository:

   ```
   git remote add origin https://github.com/yourusername/your-repo-name.git
   ```

2. Push the changes to the GitHub repository:

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

1.5.5 Step 5: Verify the Upload

1. Go to your GitHub repository URL in a web browser.
2. Verify that the `calculator.c` file is listed and accessible in the repository.

2 Symbol Mind Reading Java Application

2.1 Description

This Java AWT application is a simple graphical program that simulates a mind-reading trick. The user is prompted to think of any two-digit number, reverse the digits, and find the difference between the original and reversed numbers. The user then finds the resulting number in a grid of symbols, each labeled with a number from 0 to 98.

The twist of the program is that all numbers divisible by 9 share the same symbol, which is randomly generated each time the program runs. This symbol is eventually revealed as the "mind-read" symbol when the user clicks the `Submit` button.

2.2 Features

- **Grid of Symbols:** The main window displays a grid of 99 symbols, each paired with a number from 0 to 98.
- **Random Special Symbol:** A random symbol is assigned to all positions in the grid that are divisible by 9.
- **Instructional Message:** The application provides a brief message at the top of the window that guides the user through the mental trick.
- **Submit Button:** Once the user is ready, they click the `Submit` button to reveal the special symbol in a refreshed window.

2.3 How It Works

1. The user is instructed to think of a two-digit number, reverse its digits, and subtract the smaller number from the larger number.
2. The user then finds the result in the grid of symbols and memorizes the corresponding symbol.
3. When the user clicks the `Submit` button, the application clears the grid and displays the special symbol associated with all multiples of 9, "reading the user's mind."

2.4 How to Run

To run the program:

1. Compile the Java file using `javac SymbolApp.java`.
2. Run the compiled class using `java SymbolApp`.
3. The application window will appear, and the user can follow the on-screen instructions.

2.5 Customization

- The special symbol is generated randomly at the start of the application. You can modify the range of ASCII characters used for generating the symbol in the code if desired.
- The grid layout and other UI elements are customizable through the `GridLayout` and other layout managers used in the AWT framework.

2.5.1 Button Customization

The button has been customized as follows:

```
// Original Button Setup
submitButton = new Button("Chin Tapak Dum Dum");
submitButton.setPreferredSize(new Dimension(250, 60)); // Make the button larger
submitButton.setFont(new Font("Serif", Font.BOLD | Font.ITALIC, 20)); // Change font
submitButton.setBackground(Color.RED); // Set background color
submitButton.setForeground(Color.WHITE); // Set text color
submitButton.setCursor(new Cursor(Cursor.HAND_CURSOR)); // Change cursor when hoverin
```

These modifications include changing the button's label to `"Chin Tapak Dum Dum"`, resizing the button, adjusting the font style, and altering the button's color scheme to enhance its appearance and usability.

3 Lab Assignment 3: Git Branching, Merging, and Conflict Resolution

Task: Demonstrate proficiency in Git branching, merging, and conflict resolution in a step-by-step process.

Procedure

1. **Create a GitHub Repository:**
   - Create a new repository called `git-advanced` on GitHub.
2. **Clone the Repository:**
   - Clone the repository to your local machine using the command: `git clone <repository-url>`
3. **Create and Switch to a New Branch (feature-1):**
   - Use the command `git checkout -b feature-1` to create and switch to a new branch named `feature-1`.
4. **Add and Commit Changes on feature-1:**
   - Create a file `shared.txt` and add the content:

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

   - Stage and commit the changes: `git commit -m "Add shared.txt with original text"`
5. **Push the Branch to GitHub:**
   - Push the `feature-1` branch to GitHub: `git push origin feature-1`
6. **Create Another Branch (feature-2):**

- Switch to `feature-2` branch using the command: `git checkout -b feature-2`

7. **Modify the Shared File on feature-2:**
   - Modify the second line of `shared.txt`:

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

   - Stage and commit the changes: `git add shared.txt`
   - Commit with a descriptive message: `git commit -m "Modify Line 2 in feature-2"`
   - Push the changes: `git push origin feature-2`
8. **Switch Back to feature-1 and Modify:**
   - Switch back to `feature-1` using: `git checkout feature-1`
   - Modify the second line on `shared.txt`:

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

   - Stage and commit the changes: `git add shared.txt`
   - Commit with a descriptive message: `git commit -m "Modify Line 2 in feature-1"`
   - Push the changes: `git push origin feature-1`
9. **Merge feature-1 into main:**
   - Switch to the `main` branch: `git checkout main`
   - Merge the `feature-1` branch into the `main` branch: `git merge feature-1`
   - Push the merged changes: `git push origin main`
10. **Merge feature-2 and Handle Conflict:**
    - Merge `feature-2` into `main`: `git merge feature-2`
    - Git will notify you of a merge conflict in `shared.txt`.
    - Open `shared.txt` in your text editor and resolve the conflict by choosing the appropriate changes or combining them.
    - After resolving, stage the resolved file: `git add shared.txt`
    - Commit the merge: `git commit -m "Merge feature-2 into main and resolve conflicts"`
    - Push the resolved main branch to GitHub: `git push origin main`
11. **Clean Up Branches:**
    - Delete both `feature-1` and `feature-2` branches locally:

      ```
      git branch -d feature-1
      git branch -d feature-2
      ```

    - Delete the branches on GitHub:

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

4 LaTeX Assignments

4.1 Lab Assignment 1: Create a LaTeX Document

Task: Your task is to create a LaTeX document. The document should be formatted to look exactly like the provided attachment.

Procedure

1. **Prepare the Document:**
   - Open your LaTeX editor (Overleaf, TeXShop, or similar).
   - Create a new document and ensure it matches the formatting of the given attachment.
2. **Document Naming Convention:**
   - When your LaTeX document is complete, name it according to the following rule: `Rollno_DeptName_Firstname.tex`.
   - This file should contain your LaTeX code.
3. **Create Output and Zip Files:**
   - Compile your LaTeX document to generate the output file in PDF format.
   - Include an image file (either `.png` or `.jpg`) that should be inserted into your LaTeX document.
   - Create a zip file containing the following three files:
     - Your source code: `Rollno_DeptName_Firstname.tex`
     - Your compiled output: `Rollno_DeptName_Firstname.pdf`
     - Your image file: `.png` or `.jpg`
   - Name the zip file as `Rollno_DeptName_Firstname.zip`.

Deliverables

You need to upload the zip file containing:

- Your LaTeX source code (`.tex`)
- The compiled PDF output
- An image file (`.png` or `.jpg`)

4.2 Lab Assignment 2: Create a CV Using LaTeX

Task: Create a CV using a LaTeX document.

Procedure

1. **Outline Your CV Content:**
    - Include your name, contact details, and a professional summary.
    - List academic qualifications, work experience, skills, projects, and certifications.
2. **Decide on the Structure and Layout:**
    - Organize the CV into sections such as Personal Information, Experience, Education, etc.
3. **Choose a LaTeX Template:**
    - Select a template that suits your style from Overleaf or a LaTeX library.
4. **Customize the Template:**
    - Edit the template with your personal content (experience, qualifications, etc.).
5. **Adjust Formatting:**
    - Ensure consistency in fonts and section headings.
6. **Proofread and Finalize:**
    - Review for any errors or formatting issues.
    - Ensure alignment and organization of sections.
7. **Compile and Export:**
    - Compile the LaTeX document and export it as a PDF for sharing.

Deliverables

You need to upload the final CV in PDF format, created using LaTeX.

5 Conclusion

Throughout the lab assignments, we gained hands-on experience in several crucial aspects of software development and collaboration using Git and LaTeX. By working on tasks such as building a calculator program, modifying an existing application, handling Git branching and conflict resolution, and creating a CV using LaTeX, we strengthened our technical proficiency in C programming, version control, and document preparation.

The process of resolving merge conflicts, in particular, provided valuable insights into collaborative development and how proper Git workflows can prevent and address issues that arise when multiple team members work on the same project. Moreover, the task

of customizing a LaTeX CV highlighted the advantages of using LaTeX for professional document preparation, given its flexibility and control over formatting.

These experiences not only enhanced our understanding of the technical concepts but also fostered teamwork, problem-solving, and critical thinking, which will be beneficial for future projects and professional work environments. We look forward to applying these skills to more complex and collaborative projects in the future.