Lab Notebook

Team 38

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.:** 38
- **GitHub Repo Link:** `https://github.com/tushar02042001/Tushar-Kumar-Gupta-G-38`

Team Members

- **Member 1 (Lead):**
  - **Name: Tushar Kumar Gupta**
  - **Roll No: 30001223059**
  - **Department: BCA**
- **Member 2:**
  - **Name: Aditya Kesharwani**
  - **Roll No: 30001223028**
  - **Department: BCA**
- **Member 3:**
  - **Name: Rajdip Majumder**
  - **Roll No.: 30085323007**
  - **Department: BSC IT Cyber Security**
- **Member 4:**
  - **Name: Akash Halder**
  - **Roll No.: 30085323015**
  - **Department: BSC IT Cyber Security**
- **Member 5:**
  - **Name: Runal Saha**

  - **Roll No.: 30085323017**
  - **Department: BSC IT Cyber Security**

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 style
submitButton.setBackground(Color.RED); // Set background color
submitButton.setForeground(Color.WHITE); // Set text color
submitButton.setCursor(new Cursor(Cursor.HAND_CURSOR)); // Change cursor when hovering
```

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 Introduction

3.1 About Me

Use this paragraph to introduce yourself. You may wish to talk about your background (where you grew up, places you've traveled, your family, pets, etc.) or you could share some interesting facts about yourself or experiences you've had.

3.2 Interests & Hobbies

- Thing 1: Describe an interest or hobby.
- Thing 2: Describe an interest or hobby.
  - Include a bulleted list at least two levels deep (this is a second-level bullet).
    - This is a third-level bullet.
    - This is a third-level bullet.
  - This is a second-level bullet.
    - This is a third-level bullet.
    - This is a third-level bullet.

3.3 Favorite Quotations

1. Your favorite quote here. - Author
2. Another favorite quote here. - Author

4 Mathematics

4.1 Mathematics and Me

Reflect upon your experiences with mathematics. What do you like about mathematics? How far (if at all) would you like to take your study of mathematics? What have you enjoyed learning this year in mathematics? What have you found the most challenging?

4.2 Mathematical Notation

Choose a four-digit number which you will use to practice typesetting mathematical expressions. Typeset everything below, including all text just as you see it, substituting your four-digit number in place of the sample number 1972 wherever it occurs (use appropriate values when simplifying the equation in 4(b)).

1. Superscripts, subscripts, and Greek letters

- (a) 1972
- (b) 19^2_7
- (c) 1972
- (d) 1972
- (e) 1972π
- (f) $\cos\theta$
- (g) $\tan^{-1}(1.972)$
- (h) $\log_{19} 72$
- (i) $\ln 1972$
- (j) $e^{1.972}$
- (k) $0 < x \leq 1972$
- (l) $y \geq 1972$

2. Roots, fractions, and displaystyle

- (a) $\sqrt{1972}$
- (b) $\sqrt[19]{72}$
- (c) normal: $\frac{19}{72}$, displaystyle: $\displaystyle\frac{19}{72}$
- (d) normal: $\frac{1}{9+\frac{7}{2}}$, displaystyle: $\displaystyle\frac{1}{9+\frac{7}{2}}$
- (e) normal: $\sqrt[19]{72}$, displaystyle: $\displaystyle\sqrt[19]{72}$

3. Delimiters

- (a) display math mode:

$$\left(1+\frac{9}{72}\right)$$

- (b) display math mode:

$$\left(\frac{1}{9}-\frac{7}{2}\right)$$

4. Tables and equation arrays

- (a) Table:

x	1	2	3	4
$f(x)$	1	9	7	2

- (b) Equation Array:

$$\begin{aligned} 1+9-7\times 2 &= x \quad (1) && (1)\\ 1+9-14 &= x \quad (2) && (2)\\ 10-14 &= x \quad (3) && (3)\\ x &= -4 \quad (4) && (4) \end{aligned}$$

5. Functions & Formulas

- (a) The quadratic formula:

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

- (b) The function $f(x)=\frac{x+1}{9^2-7^2}$ has domain $D_f:(-\infty,\infty)$ and range $R_f:\left(-\frac{7}{2},\infty\right)$.
- (c) Definition of a Derivative:

$$\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}=f'(x)$$

- (d) Chain Rule: $[f(g(x))]'=f'(g(x))\cdot g'(x)$
- (e) $\frac{d^2y}{dx^2}=f''(x)$
- (f) $\int \sec^2 x\,dx=\tan x+C$
- (g) $\int e^{2x}\,dx=\frac{1}{2}e^{2x}+C$
- (h) Fundamental Theorem of Calculus, Part 1: $\int_a^b f'(x)\,dx=f(b)-f(a)$
- (i) Fundamental Theorem of Calculus, Part 2: $\frac{d}{dx}\left(\int_a^{g(x)} f(t)\,dt\right)=f(g(x))\cdot g'(x)$
- (j) Euler's Method: $y_1=y_0+hF(x_0,y_0)$ where h is the step size, and $F(x,y)=\frac{dy}{dx}$

- (k) $a_n = 1972, \frac{1972}{2}, \frac{1972}{2^2}, \frac{1972}{2^3}, \ldots, \frac{1972}{2^n}$ represents a geometric sequence.
- (l) $S_n = \sum_{n=1}^{\infty} \frac{1972}{2^n}$ is a convergent geometric series since $|r| = \frac{1}{2} < 1$.
- (m) Taylor Series: $\sum_{n=0}^{\infty} \frac{f^{(n)}(c)}{n!}(x-c)^n$
- (n) Velocity Vector: $\vec{v}(t) = x'(t)\hat{\imath} + y'(t)\hat{\jmath} = \left(\frac{dx}{dt}, \frac{dy}{dt}\right)$
- (o) Area of Polar Curve: $A = \frac{1}{2}\int_{\alpha}^{\beta} r^2\, d\theta$

5 Lab Assignment: Calculator Program using C

5.1 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.

5.2 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.

5.3 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;
}
```

5.4 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
   ```

5.5 Adding the Calculator Program to GitHub Repository

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

5.5.1 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.

5.5.2 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.

5.5.3 Step 3: Commit the Changes

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

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

5.5.4 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
```

5.5.5 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.