Experiment – 1 a: TypeScript

Name of Student	SHRAVANI S RASAM
Class Roll No	D15A 45
D.O.P.	06\02\2025
D.O.S.	
Sign and Grade	

Experiment – 1 a: TypeScript

1. **Aim:** Write a simple TypeScript program using basic data types (number, string, boolean) and operators.
2. **Problem Statement:**

   a. Create a calculator in TypeScript that uses basic operations like addition, subtraction, multiplication, and division. It also gracefully handles invalid operations and division by zero..

   b. Design a Student Result database management system using TypeScript.

Theory:

1. What are the different data types in TypeScript?

- **Primitive Types**: number, string, boolean, null, undefined, symbol, bigint
- **Array**: number[], Array<number>
- **Tuple**: [string, number]
- **Enum**: enum Color {Red, Green, Blue}
- **Any**: any
- **Void**: No return type for functions
- **Never**: Represents values that never occur

What are Type Annotations in TypeScript?

Type annotations define the type of variables, function parameters, or return values to ensure type safety and avoid errors at compile time.

typescript

```
let x: number = 5;

function greet(name: string): string { return "Hello " + name; }
```

2. How do you compile TypeScript files?

Use the TypeScript compiler (tsc) to compile .ts files into .js files.

```
sc filename.ts
```

3. What is the difference between JavaScript and TypeScript?

- **JavaScript**: Dynamically typed, interpreted language.
- **TypeScript**: Statically typed superset of JavaScript that is compiled into JavaScript.

4. How do JavaScript and TypeScript implement Inheritance?

- **JavaScript**: Uses prototype-based inheritance.
- **TypeScript**: Uses class-based inheritance with the extends keyword.

Example:

JavaScript:
javascript
Copy

```
function Animal(name) { this.name = name; }

Animal.prototype.speak = function() { console.log(this.name); };
```

TypeScript:

```
class Animal {

  constructor(public name: string) {}

  speak() { console.log(this.name); }

}

class Dog extends Animal { }
```

5. How do generics make the code flexible and why should we use generics over other types?

Generics allow functions, classes, and interfaces to work with any data type while maintaining type safety. Using generics avoids the risks associated with any, which doesn't provide type checking, preventing potential runtime errors.

In the lab assignment 3, using generics ensures type safety while handling various input types, unlike any, which skips type checking.

6. What is the difference between Classes and Interfaces in TypeScript? Where are interfaces used?

- **Classes**: Define objects with both implementation (methods and properties).
- **Interfaces**: Define the structure (shape) of an object without implementation.

Interfaces Usage: Used to enforce structure in objects, function signatures, and class contracts.

Output:

A.

CODE

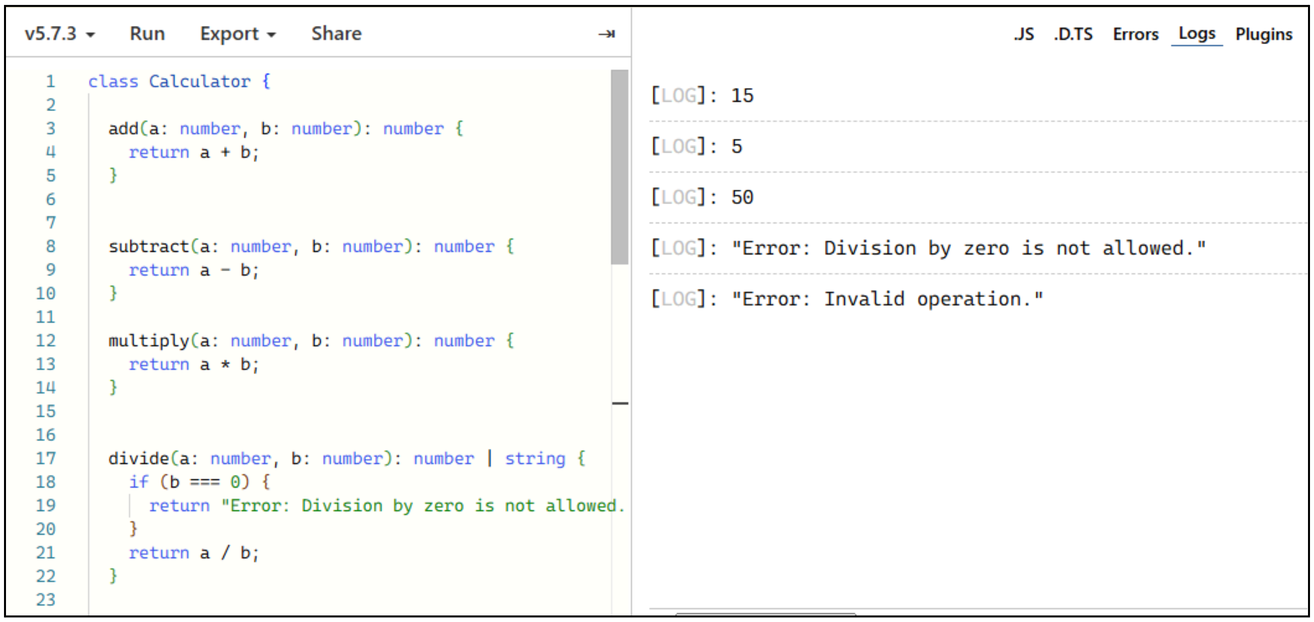
Ts File

```
class Calculator {

  add(a: number, b: number): number {
    return a + b;
  }
subtract(a: number, b: number): number {
    return a - b;
  }
  multiply(a: number, b: number): number {
    return a * b;
  }
  divide(a: number, b: number): number | string {
    if (b === 0) {
      return "Error: Division by zero is not allowed.";
    }
    return a / b;
  }

  operate(a: number, b: number, operation: string): number | string {
    switch (operation) {
      case "add":
        return this.add(a, b);
      case "subtract":
        return this.subtract(a, b);
      case "multiply":
        return this.multiply(a, b);
      case "divide":
        return this.divide(a, b);
      default:
        return "Error: Invalid operation.";
        }}}

        // Create an instance of the Calculator class

        const calc = new Calculator();


        console.log(calc.operate(10, 5, "add"));

        console.log(calc.operate(10, 5, "subtract"));

        console.log(calc.operate(10, 5, "multiply"));

        console.log(calc.operate(10, 0, "divide"));

        console.log(calc.operate(10, 5, "modulus"));
```

Js File

```
"use strict";
class Calculator {
    add(a, b) {
        return a + b;
    }
    subtract(a, b) {
        return a - b;
    }
    multiply(a, b) {
        return a * b;
    }
    divide(a, b) {
        if (b === 0) {
            return "Error: Division by zero is not allowed.";
        }
        return a / b;
    }
    operate(a, b, operation) {
        switch (operation) {
            case "add":
                return this.add(a, b);
            case "subtract":
                return this.subtract(a, b);
            case "multiply":
                return this.multiply(a, b);
            case "divide":
                return this.divide(a, b);
            default:
                return "Error: Invalid operation.";
        }
    }
}
// Create an instance of the Calculator class
const calc = new Calculator();

console.log(calc.operate(10, 5, "add"));
console.log(calc.operate(10, 5, "subtract"));
console.log(calc.operate(10, 5, "multiply"));
console.log(calc.operate(10, 0, "divide"));
console.log(calc.operate(10, 5, "modulus"));
```

OUTPUT:



```
class Calculator {

  add(a: number, b: number): number {
    return a + b;
  }


  subtract(a: number, b: number): number {
    return a - b;
  }

  multiply(a: number, b: number): number {
    return a * b;
  }


  divide(a: number, b: number): number | string {
    if (b === 0) {
      return "Error: Division by zero is not allowed.
    }
    return a / b;
  }

```

```
[LOG]: 15
[LOG]: 5
[LOG]: 50
[LOG]: "Error: Division by zero is not allowed."
[LOG]: "Error: Invalid operation."
```

B

CODE:

Ts File

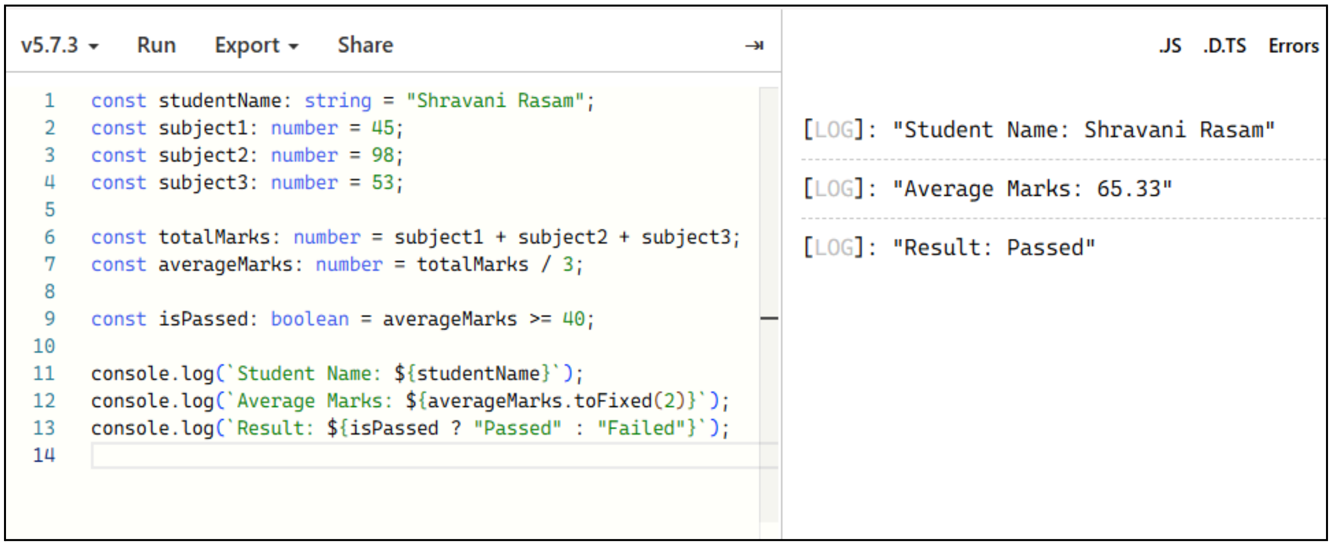
```
const studentName: string = "Shravani Rasam";

const subject1: number = 45;

const subject2: number = 98;

const subject3: number = 53;



const totalMarks: number = subject1 + subject2 + subject3;

const averageMarks: number = totalMarks / 3;



const isPassed: boolean = averageMarks >= 40;



console.log(`Student Name: ${studentName}`);

console.log(`Average Marks: ${averageMarks.toFixed(2)}`);

console.log(`Result: ${isPassed ? "Passed" : "Failed"}`);
```

Js File

```
"use strict";
const studentName = "Shravani Rasam";
const subject1 = 45;
const subject2 = 98;
const subject3 = 53;
const totalMarks = subject1 + subject2 + subject3;
const averageMarks = totalMarks / 3;
const isPassed = averageMarks >= 40;
console.log(`Student Name: ${studentName}`);
console.log(`Average Marks: ${averageMarks.toFixed(2)}`);
```

console.log(`Result: ${isPassed ? "Passed" : "Failed"}`);

OUTPUT:



```
const studentName: string = "Shravani Rasam";
const subject1: number = 45;
const subject2: number = 98;
const subject3: number = 53;

const totalMarks: number = subject1 + subject2 + subject3;
const averageMarks: number = totalMarks / 3;

const isPassed: boolean = averageMarks >= 40;

console.log(`Student Name: ${studentName}`);
console.log(`Average Marks: ${averageMarks.toFixed(2)}`);
console.log(`Result: ${isPassed ? "Passed" : "Failed"}`);
```

```
[LOG]: "Student Name: Shravani Rasam"
[LOG]: "Average Marks: 65.33"
[LOG]: "Result: Passed"
```