Experiment – 1 a: TypeScript

Name of Student	Jai Talreja
Class Roll No	59
D.O.P.	
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.

```
// Step 1: Declare basic data types
const studentName: string = "John Doe";
const subject1: number = 45;
const subject2: number = 38;
const subject3: number = 50;

// Step 2: Calculate the average marks
const totalMarks: number = subject1 + subject2 + subject3;
const averageMarks: number = totalMarks / 3;

// Step 3: Determine if the student has passed or failed
const isPassed: boolean = averageMarks >= 40;

// Step 4: Display the result
console.log(Student Name: ${studentName});
```

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

3. **Theory:**
    a. What are the different data types in TypeScript? What are Type Annotations in Typescript?
    b. How do you compile TypeScript files?
    c. What is the difference between JavaScript and TypeScript?
    d. Compare how Javascript and Typescript implement Inheritance.
    e. How generics make the code flexible and why we should use generics over other types. In the lab assignment 3, why the usage of generics is more suitable than using any data type to handle the input.
    f. What is the difference between Classes and Interfaces in Typescript? Where are interfaces used?

4. **Output:**

[a. Calculator]

```
class Calculator {

  // Method for addition

  static add(a: number, b: number): number {

    return a + b;

  }


  // Method for subtraction

  static subtract(a: number, b: number): number {

    return a - b;

  }
```

```
// Method for multiplication
static multiply(a: number, b: number): number {
  return a * b;
}

// Method for division with error handling for division by zero
static divide(a: number, b: number): number | string {
  if (b === 0) {
    return 'Error: Division by zero is not allowed';
  }
  return a / b;
}

// Method to handle invalid operation
static calculate(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';
    }
  }
}

// Example Usage
const num1 = 10;
const num2 = 5;
console.log(Calculator.calculate(num1, num2, 'add')); // 15
console.log(Calculator.calculate(num1, num2, 'subtract')); // 5
console.log(Calculator.calculate(num1, num2, 'multiply')); // 50
console.log(Calculator.calculate(num1, num2, 'divide')); // 2
console.log(Calculator.calculate(num1, 0, 'divide')); // Error: Division by zero is not allowed
console.log(Calculator.calculate(num1, num2, 'modulus')); // Error: Invalid operation
//Program ends here.
```

```
jai-talreja@hp-envy-x360:~/Documents/WebX$ ls
Exp_1a_a.ts  tsconfig.json
jai-talreja@hp-envy-x360:~/Documents/WebX$ tsc Exp_1a_a.ts
jai-talreja@hp-envy-x360:~/Documents/WebX$ ls
Exp_1a_a.js  Exp_1a_a.ts  tsconfig.json
jai-talreja@hp-envy-x360:~/Documents/WebX$ tsc
jai-talreja@hp-envy-x360:~/Documents/WebX$ ls
Exp_1a_a.js  Exp_1a_a.ts  tsconfig.json
jai-talreja@hp-envy-x360:~/Documents/WebX$
jai-talreja@hp-envy-x360:~/Documents/WebX$ ls
Exp_1a_a.js  Exp_1a_a.ts  tsconfig.json
jai-talreja@hp-envy-x360:~/Documents/WebX$ node Exp_1a_a.js
15
5
50
2
Error: Division by zero is not allowed
Error: Invalid operation
jai-talreja@hp-envy-x360:~/Documents/WebX$
```

[b. Student result Database]

```
class Student {
    studentName: string;
    subjectMarks: number[];

    constructor(studentName: string, subjectMarks: number[]) {
        this.studentName = studentName;
        this.subjectMarks = subjectMarks;
    }

    // Step 3: Calculate the total marks
    getTotalMarks(): number {
        return this.subjectMarks.reduce((acc, mark) => acc + mark, 0);
    }

    // Step 4: Calculate the average marks
    getAverageMarks(): number {
        return this.getTotalMarks() / this.subjectMarks.length;
    }

    // Step 5: Determine if the student has passed or failed
    isPassed(): boolean {
        const average = this.getAverageMarks();
        return average >= 40;
```

```
    }

    // Step 6: Display the student result
    displayResult(): void {
        console.log(`Student Name: ${this.studentName}`);
        console.log(`Average Marks: ${this.getAverageMarks()}`);
        console.log(`Result: ${this.isPassed() ? "Passed" : "Failed"}`);
    }
}

// Example usage
const student1 = new Student("John Doe", [45, 38, 50]);
const student2 = new Student("Jane Smith", [75, 82, 65]);

student1.displayResult();
student2.displayResult();
//Program ends here.
```

```
jai-talreja@hp-envy-x360:~/Documents/WebX$ ls
Exp_1a_a.js  Exp_1a_a.ts  Exp_1a_b.ts  tsconfig.json
jai-talreja@hp-envy-x360:~/Documents/WebX$ tsc --init
error TS5054: A 'tsconfig.json' file is already defined at: '/home/jai-talreja/Documents/WebX/tsconfig.json'.
jai-talreja@hp-envy-x360:~/Documents/WebX$ ls
Exp_1a_a.js  Exp_1a_a.ts  Exp_1a_b.ts  tsconfig.json
jai-talreja@hp-envy-x360:~/Documents/WebX$ tsc Exp_1a_b.ts
jai-talreja@hp-envy-x360:~/Documents/WebX$ ls
Exp_1a_a.js  Exp_1a_a.ts  Exp_1a_b.js  Exp_1a_b.ts  tsconfig.json
jai-talreja@hp-envy-x360:~/Documents/WebX$ node Exp_1a_b.js
Student Name: John Doe
Average Marks: 44.333333333333336
Result: Passed
Student Name: Jane Smith
Average Marks: 74
Result: Passed
jai-talreja@hp-envy-x360:~/Documents/WebX$
```