TYPESCRIPT PROGRAMMING

LAB EXPERIMENT

- **Experiment 1: Print Hello World**

Aim:

To print “Hello World” using TypeScript.

Input:

No input

Output:

Hello World

Program:

```
console.log("Hello World");
```

- **Experiment 2: Sum of Two Numbers**

Aim:

To find the sum of two numbers.

Input:

10

20

Output:

30

Program:

```
let a: number = 10;

let b: number = 20;


let sum: number = a + b;

console.log(sum);
```

- **Experiment 3: Difference of Two Numbers**

Aim:

To find the difference of two numbers.

Input:

20

5

Output:

15

Program:

```
let a: number = 20;
let b: number = 5;


console.log(a - b);
```

- **Experiment 4: Check Even or Odd**

Aim:

To check whether a number is even or odd.

Input:

7

Output:

Odd

Program:

```
let n: number = 7;
if (n % 2 === 0) {
   console.log("Even");
} else {
   console.log("Odd");
}
```

- **Experiment 5: Positive, Negative or Zero**

Aim:

To check whether a number is positive, negative or zero.

Input:

-3

Output:

Negative

Program:

```
let n: number = -3;


if (n > 0) {
   console.log("Positive");
} else if (n < 0) {
   console.log("Negative");
} else {
   console.log("Zero");
}
```

- **Experiment 6: Largest of Two Numbers**

Aim:

To find the largest of two numbers using TypeScript.

Input:

12

20

Output:

20

Program:

```
let a: number = 12;
let b: number = 20;
```

```
if (a > b) {
   console.log(a);
} else {
   console.log(b);
}
```

- **Experiment 7: Largest of Three Numbers**

Aim:

To find the largest among three numbers.

Input:

5

9

3

Output:

9

Program:

```
let a: number = 5;
let b: number = 9;
let c: number = 3;

if (a >= b && a >= c) {
   console.log(a);
} else if (b >= a && b >= c) {
   console.log(b);
} else {
   console.log(c);
}
```

- **Experiment 8: Swap Two Numbers**

Aim:

To swap two numbers using a temporary variable.

Input:

10

20

Output:

After Swapping: 20 10

Program:

```
let a: number = 10;
let b: number = 20;

let temp: number = a;
a = b;
b = temp;

console.log("After Swapping:", a, b);
```

- **Experiment 9: Simple Interest Calculation**

Aim:

To calculate simple interest.

Formula:

SI = (P × R × T) / 100

Input:

1000

5

2

Output:

100

Program:

```
let p: number = 1000;
let r: number = 5;
let t: number = 2;

let si: number = (p * r * t) / 100;
console.log(si);
```

- **Experiment 10: Area of a Circle**

Aim:

To calculate the area of a circle.

Formula:

Area = π × r × r

Input:

7

Output:

153.94

Program:

```
let r: number = 7;
let area: number = 3.14159 * r * r;
console.log(area.toFixed(2));
```

- **Experiment 11: Leap Year Check**

Aim:

To check whether a given year is a leap year.

Input:

2024

Output:

Leap Year

Program:

```
let year: number = 2024;

if ((year % 4 === 0 && year % 100 !== 0) || year % 400 === 0) {
    console.log("Leap Year");
} else {
    console.log("Not a Leap Year");
}
```

- **Experiment 12: Factorial of a Number**

Aim:

To find the factorial of a given number.

Input:

5

Output:

120

Program:

```
let n: number = 5;
let fact: number = 1;
for (let i = 1; i <= n; i++) {
    fact *= i;
}
console.log(fact);
```

- **Experiment 13: Fibonacci Series**

Aim:

To print the Fibonacci series up to N terms.

Input:

5

Output:

0 1 1 2 3

Program:

```
let n: number = 5;
let a: number = 0;
let b: number = 1;
let result: string = a + " " + b;
for (let i = 3; i <= n; i++) {
    let c: number = a + b;
    result += " " + c;
    a = b;
    b = c;
}
console.log(result);
```

- **Experiment 14: Reverse a Number**

Aim:

To reverse a given number.

Input:

1204

Output:

4021

Program:

```
let n: number = 1204;
let rev: number = 0;
while (n !== 0) {
    rev = rev * 10 + (n % 10);
    n = Math.floor(n / 10);
}console.log(rev);
```

Experiment 15: Palindrome Number

Aim:

To check whether a given number is a palindrome.

Input:

121

Output:

Palindrome

Program:

```
let num: number = 121;
let temp: number = num;
let rev: number = 0;
while (num !== 0) {
    rev = rev * 10 + (num % 10);
    num = Math.floor(num / 10);
}
if (rev === temp) {
    console.log("Palindrome");
} else {
    console.log("Not Palindrome");
}
```

Experiment 16: Sum of Digits

Aim:

To find the sum of digits of a given number.

Input:

123

Output:

6

Program:

```
let n: number = 123;
let sum: number = 0;

while (n !== 0) {
    sum += n % 10;
    n = Math.floor(n / 10);
}

console.log(sum);
```

- **Experiment 17: Count Number of Digits**

Aim:

To count the number of digits in a given number.

Input:

4589

Output:

4

Program:

```
let n: number = 4589;
let count: number = 0;

while (n !== 0) {
    count++;
    n = Math.floor(n / 10);
}

console.log(count);
```

- **Experiment 18: Multiplication Table**

Aim:

To print the multiplication table of a given number.

Input:

5

Output:

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

Program:

```
let n: number = 5;


for (let i = 1; i <= 10; i++) {
    console.log(n + " x " + i + " = " + (n * i));
}
```

- **Experiment 19: Prime Number Check**

Aim:

To check whether a given number is prime.

Input:

7

Output:

Prime

Program:

```
let n: number = 7;

let isPrime: boolean = true;
```

```
if (n <= 1) {
    isPrime = false;
}

for (let i = 2; i <= Math.floor(n / 2); i++) {
    if (n % i === 0) {
        isPrime = false;
        break;
    }
}

if (isPrime) {
    console.log("Prime");
} else {
    console.log("Not Prime");
}
```

- **Experiment 20: Prime Numbers Between a Range**

Aim:

To print all prime numbers between a given range.

Input:

10

30

Output:

11 13 17 19 23 29

Program:

```
let start: number = 10;
let end: number = 30;
let result: string = "";
```

```
for (let i = start; i <= end; i++) {
    if (i <= 1) continue;
    let prime: boolean = true;

    for (let j = 2; j <= Math.floor(i / 2); j++) {
        if (i % j === 0) {
            prime = false;
            break;
        }
    }
    if (prime) {
        result += i + " ";
    }
}
```

- **Experiment 21: Read and Display Array Elements**

Aim:

To read elements into an array and display them.

Input:

5

10 20 30 40 50

Output:

10 20 30 40 50

Program:

```
let arr: number[] = [10, 20, 30, 40, 50];
let result: string = "";

for (let i = 0; i < arr.length; i++) {
```

```
    result += arr[i] + " ";
}

console.log(result.trim());
```

- **Experiment 22: Sum and Average of Array Elements**

Aim:

To find the sum and average of elements in an array.

Input:

4

10 20 30 40

Output:

Sum = 100

Average = 25

Program:

```
let arr: number[] = [10, 20, 30, 40];
let sum: number = 0;
for (let i = 0; i < arr.length; i++)
{
    sum += arr[i];
}

let avg: number = sum / arr.length;
console.log("Sum = " + sum);
console.log("Average = " + avg);
```

- **Experiment 23: Largest Element in an Array**

Aim:

To find the largest element in an array.

Input:

5

2 8 1 6 4

Output:

8

Program:

```
let arr: number[] = [2, 8, 1, 6, 4];
let max: number = arr[0];

for (let i = 1; i < arr.length; i++) {
   if (arr[i] > max) {
      max = arr[i];
   }
}

console.log(max);
```

- **Experiment 24: Smallest Element in an Array**

Aim:

To find the smallest element in an array.

Input:

5

2 8 1 6 4

Output:

1

Program:

```
let arr: number[] = [2, 8, 1, 6, 4];
let min: number = arr[0];
```

```
for (let i = 1; i < arr.length; i++) {
    if (arr[i] < min) {
        min = arr[i];
    }
}

console.log(min);
```

- **Experiment 25: Linear Search in an Array**

Aim:

To search an element in an array using linear search.

Input:

5

10 20 30 40 50

30

Output:

Element found at position 3

Program:

```
let arr: number[] = [10, 20, 30, 40, 50];
let key: number = 30;
let found: boolean = false;

for (let i = 0; i < arr.length; i++) {
    if (arr[i] === key) {
        console.log("Element found at position " + (i + 1));
        found = true;
        break;
    }
}
```

```
if (!found) {
    console.log("Element not found");
}
```

🔹 Experiment 26: Binary Search in an Array

Aim:

To search an element in a sorted array using binary search.

Input:

5

10 20 30 40 50

30

Output:

Element found at position 3

Program:

```
let arr: number[] = [10, 20, 30, 40, 50];
let key: number = 30;

let low: number = 0;
let high: number = arr.length - 1;
let found: boolean = false;

while (low <= high) {
    let mid: number = Math.floor((low + high) / 2);

    if (arr[mid] === key) {
        console.log("Element found at position " + (mid + 1));
        found = true;
        break;
```

```
    } else if (arr[mid] < key) {
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}

if (!found) {
    console.log("Element not found");
}
```

- **Experiment 27: Reverse an Array**

Aim:

To reverse the elements of an array.

Input:

5

1 2 3 4 5

Output:

5 4 3 2 1

Program:

```
let arr: number[] = [1, 2, 3, 4, 5];
let n: number = arr.length;
for (let i = 0; i < n / 2; i++)
{
    let temp: number = arr[i];
    arr[i] = arr[n - i - 1];
    arr[n - i - 1] = temp;
}
console.log(arr.join(" "));
```

Experiment 28: Sort an Array in Ascending Order

Aim:

To sort the elements of an array in ascending order.

Input:

5

4 2 5 1 3

Output:

1 2 3 4 5

Program:

```
let arr: number[] = [4, 2, 5, 1, 3];


for (let i = 0; i < arr.length - 1; i++) {
    for (let j = i + 1; j < arr.length; j++) {
        if (arr[i] > arr[j]) {
            let temp: number = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
}


console.log(arr.join(" "));
```

Experiment 29: Sort an Array in Descending Order

Aim:

To sort the elements of an array in descending order.

Input:

5

4 2 5 1 3

Output:

5 4 3 2 1

Program:

```
let arr: number[] = [4, 2, 5, 1, 3];


for (let i = 0; i < arr.length - 1; i++) {
   for (let j = i + 1; j < arr.length; j++) {
      if (arr[i] < arr[j]) {
         let temp: number = arr[i];
         arr[i] = arr[j];
         arr[j] = temp;
      }
   }
}


console.log(arr.join(" "));
```

- **Experiment 30: Frequency of Elements in an Array**

Aim:

To find the frequency of each element in an array.

Input:

5

1 2 2 3 1

Output:

1 -> 2

2 -> 2

3 -> 1

Program:

```
let arr: number[] = [1, 2, 2, 3, 1];
let visited: boolean[] = new Array(arr.length).fill(false);

for (let i = 0; i < arr.length; i++) {
    if (visited[i]) continue;

    let count: number = 1;
    for (let j = i + 1; j < arr.length; j++) {
        if (arr[i] === arr[j]) {
            visited[j] = true;
            count++;
        }
    }

    console.log(arr[i] + " -> " + count);
}
```

- **Experiment 31: Class and Object**

Aim:

To demonstrate the concept of class and object in TypeScript.

Input:

No input

Output:

Name: Ravi

Age: 20

Program:

```
class Student {
    name: string;
```

```
    age: number;

    constructor(name: string, age: number) {
        this.name = name;
        this.age = age;
    }
}

let s = new Student("Ravi", 20);
console.log("Name:", s.name);
console.log("Age:", s.age);
```

- **Experiment 32: Constructor Demonstration**

Aim:

To demonstrate the use of a constructor in TypeScript.

Input:

No input

Output:

Constructor Called

Program:

```
class Demo {
    constructor() {
        console.log("Constructor Called");
    }
}

let d = new Demo();
```

🔹 Experiment 33: Method Overloading (Using Optional Parameters)

Aim:

To demonstrate method overloading in TypeScript using optional parameters.

Input:

10 20

5 6 7

Output:

30

18

Program:

```
class Add {
   sum(a: number, b: number, c?: number): number {
      if (c !== undefined) {
         return a + b + c;
      }
      return a + b;
   }}
let obj = new Add();
console.log(obj.sum(10, 20));
console.log(obj.sum(5, 6, 7));
```

🔹 Experiment 34: Inheritance

Aim:

To demonstrate inheritance in TypeScript.

Input:

No input

Output:

This is Parent class

This is Child class

Program:

```
class Parent {
    showParent(): void {
        console.log("This is Parent class");
    }
}

class Child extends Parent {
    showChild(): void {
        console.log("This is Child class");
    }
}

let c = new Child();
c.showParent();
c.showChild();
```

- **Experiment 35: Method Overriding**

Aim:

To demonstrate method overriding in TypeScript.

Input:

No input

Output:

This is Parent method

This is Child method

Program:

```
class Parent {
    show(): void {
        console.log("This is Parent method");
```

```
    }
}

class Child extends Parent {
    show(): void {
        console.log("This is Child method");
    }
}

let p1 = new Parent();
p1.show();

let p2: Parent = new Child();
p2.show();
```

- **Experiment 36: Abstract Class**

Aim:

To demonstrate the use of an abstract class in TypeScript.

Input:

No input

Output:

Drawing Circle

Program:

```
abstract class Shape {
    abstract draw(): void;
}

class Circle extends Shape {
    draw(): void {
```

```
        console.log("Drawing Circle");
    }
}

let s: Shape = new Circle();
s.draw();
```

- **Experiment 37: Interface Implementation**

Aim:

To demonstrate interface implementation in TypeScript.

Input:

No input

Output:

Interface method implemented

Program:

```
interface DemoInterface {
    show(): void;
}

class DemoClass implements DemoInterface {
    show(): void {
        console.log("Interface method implemented");
    }
}

let d: DemoInterface = new DemoClass();
d.show();
```

- **Experiment 38: Encapsulation (Getter and Setter)**

Aim:

To demonstrate encapsulation using getter and setter methods.

Input:

101

Arun

Output:

ID: 101

Name: Arun

Program:

```
class Student {
    private id: number = 0;
    private name: string = "";

    setId(id: number): void {
        this.id = id;
    }

    setName(name: string): void {
        this.name = name;
    }

    getId(): number {
        return this.id;
    }

    getName(): string {
        return this.name;
    }
```

```
}

let s = new Student();
s.setId(101);
s.setName("Arun");

console.log("ID:", s.getId());
console.log("Name:", s.getName());
```

- **Experiment 39: Access Modifiers**

Aim:

To demonstrate access modifiers in TypeScript.

Input:

No input

Output:

Public Variable

Protected Variable

Program:

```
class Demo {
    public a: string = "Public Variable";
    protected b: string = "Protected Variable";
}

class Test extends Demo {
    show(): void {
        console.log(this.a);
        console.log(this.b);
    }
}
```

```
let t = new Test();
t.show();
```

Experiment 40: Static Members

Aim:

To demonstrate static members in TypeScript.

Input:

No input

Output:

Count = 2

Program:

```
class Counter {
    static count: number = 0;

    constructor() {
        Counter.count++;
    }

    static showCount(): void {
        console.log("Count =", Counter.count);
    }
}

new Counter();
new Counter();
Counter.showCount();
```

- **Experiment 41: Reverse a String**

Aim:

To reverse a given string using TypeScript.

Input:

programming

Output:

gnimmargorp

Program:

```
let str: string = "programming";
let rev: string = "";

for (let i = str.length - 1; i >= 0; i--) {
    rev += str[i];
}
console.log(rev);
```

- **Experiment 42: String Palindrome**

Aim:

To check whether a given string is a palindrome.

Input:

madam

Output:

Palindrome

Program:

```
let str: string = "madam";
let rev: string = "";

for (let i = str.length - 1; i >= 0; i--) {
    rev += str[i];
```

```
}

if (str === rev) {
   console.log("Palindrome");
} else {
   console.log("Not Palindrome");
}
```

- **Experiment 43: Count Vowels and Consonants**

Aim:

To count the number of vowels and consonants in a string.

Input:

hello

Output:

Vowels: 2

Consonants: 3

Program:

```
let str: string = "hello".toLowerCase();
let vowels: number = 0;
let consonants: number = 0;

for (let ch of str) {
   if (ch >= 'a' && ch <= 'z') {
      if ("aeiou".includes(ch)) {
         vowels++;
      } else {
         consonants++;
      }
   }
```

```
}

console.log("Vowels:", vowels);
console.log("Consonants:", consonants);
```

- **Experiment 44: String Comparison**

Aim:

To compare two strings.

Input:

Hello

Hello

Output:

Strings are Equal

Program:

```
let s1: string = "Hello";
let s2: string = "Hello";

if (s1 === s2) {
    console.log("Strings are Equal");
} else {
    console.log("Strings are Not Equal");
}
```

- **Experiment 45: Error Handling (Try–Catch)**

Aim:

To demonstrate error handling using try–catch.

Input:

10

0

Output:

Error occurred

Program:

```
try {
    let a: number = 10;
    let b: number = 0;

    if (b === 0) {
        throw new Error("Division by zero");
    }

    console.log(a / b);
} catch (error) {
    console.log("Error occurred");
}
```

- **Experiment 46: Custom Error Class**

Aim:

To demonstrate a user-defined error in TypeScript.

Input:

15

Output:

Not Eligible

Program:

```
class AgeError extends Error {
    constructor(message: string) {
        super(message);
    }
}
```

```
let age: number = 15;


try {
   if (age < 18) {
      throw new AgeError("Not Eligible");
   }
   console.log("Eligible");
} catch (e: any) {
   console.log(e.message);
}
```

- **Experiment 47: Write Data to a File (Node.js)**

Aim:

To write data into a file using TypeScript.

Input:

Hello TypeScript

Output:

Data written successfully

Program:

```
import * as fs from "fs";
fs.writeFileSync("data.txt", "Hello TypeScript");
console.log("Data written successfully");
```

- **Experiment 48: Read Data from a File (Node.js)**

Aim:

To read data from a file using TypeScript.

Input:

data.txt

Output:

Hello TypeScript

Program:

```
import * as fs from "fs";

let data: string = fs.readFileSync("data.txt", "utf-8");
console.log(data);
```

- **Experiment 49: Async / Await**

Aim:

To demonstrate async and await in TypeScript.

Input:

No input

Output:

Async Task Completed

Program:

```
async function demoAsync(): Promise<void> {
   return new Promise((resolve) => {
      setTimeout(() => {
         console.log("Async Task Completed");
         resolve();
      }, 1000);
   });
}

demoAsync();
```

- **Experiment 50: Menu-Driven Program**

Aim:

To create a menu-driven program using switch case.

Input:

1

10

20

Output:

Sum = 30

Program:

```
let choice: number = 1;
let a: number = 10;
let b: number = 20;

switch (choice) {
    case 1:
        console.log("Sum =", a + b);
        break;
    case 2:
        console.log("Difference =", a - b);
        break;
    default:
        console.log("Invalid Choice");
}
```

ANALYTICAL THINKING PROGRAMS

Experiment 1: Password Length Validation

Aim:

To check whether a password is strong based on length.

Input:

mypassword

Output:

Strong Password

Program:

```
let password: string = "mypassword";
if (password.length >= 8)
   console.log("Strong Password");
else
   console.log("Weak Password");
```

Experiment 2: Username Space Validation

Aim:

To validate username by checking spaces.

Input:

user name

Output:

Invalid Username

Program:

```
let username: string = "user name";
if (username.includes(" "))
   console.log("Invalid Username");
else
   console.log("Valid Username");
```

Experiment 3: Email Validation

Aim:

To check whether an email contains '@'.

Input:

user@gmail.com

Output:

Valid Email

Program:

```
let email: string = "user@gmail.com";


if (email.includes("@"))
   console.log("Valid Email");
else
   console.log("Invalid Email");
```

Experiment 4: Dark Mode Detection

Aim:

To display application mode.

Input:

dark

Output:

Dark Mode Enabled

Program:

```
let mode: string = "dark";


if (mode === "dark")
   console.log("Dark Mode Enabled");
else
   console.log("Light Mode Enabled");
```

Experiment 5: Cart Empty Check

Aim:

To check whether shopping cart is empty.

Input:

[]

Output:

Cart is Empty

Program:

```
let cart: string[] = [];


if (cart.length === 0)
   console.log("Cart is Empty");
else
   console.log("Cart has items");
```

Experiment 6: Login Attempt Lock

Aim:

To lock account after multiple login attempts.

Input:

3

Output:

Account Locked

Program:

```
let attempts: number = 3;
if (attempts >= 3)
   console.log("Account Locked");
```

Experiment 7: Phone Number Length Validation

Aim:

To validate phone number length.

Input:

9876543210

Output:

Valid Number

Program:

```
let phone: string = "9876543210";

if (phone.length === 10)
   console.log("Valid Number");
else
   console.log("Invalid Number");
```

Experiment 8: File Size Validation

Aim:

To check file upload size.

Input:

6

Output:

File Too Large

Program:

```
let fileSize: number = 6;

if (fileSize > 5)
   console.log("File Too Large");
else
   console.log("Upload Successful");
```

Experiment 9: Internet Status Check

Aim:

To display internet connection status.

Input:

false

Output:

Offline

Program:

```
let isOnline: boolean = false;


if (isOnline)
   console.log("Online");
else
   console.log("Offline");
```

Experiment 10: User Role Access

Aim:

To check access level based on role.

Input:

admin

Output:

Full Access

Program:

```
let role: string = "admin";


if (role === "admin")
   console.log("Full Access");
else
   console.log("Limited Access");
```

Experiment 11: Marks Range Validation

Aim:

To validate marks input range.

Input:

105

Output:

Invalid Marks

Program:

```
let marks: number = 105;


if (marks < 0 || marks > 100)
   console.log("Invalid Marks");
else
   console.log("Valid Marks");
```

Experiment 12: Language Selection

Aim:

To display selected language.

Input:

ta

Output:

Tamil

Program:

```
let lang: string = "ta";
switch (lang) {
   case "en": console.log("English"); break;
   case "ta": console.log("Tamil"); break;
   default: console.log("Other Language");
}
```

Experiment 13: Stock Availability

Aim:

To check product stock availability.

Input:

0

Output:

Out of Stock

Program:

```
let stock: number = 0;


console.log(stock > 0 ? "Available" : "Out of Stock");
```

Experiment 14: Password Match Check

Aim:

To verify password confirmation.

Input:

abc123

abc123

Output:

Password Matched

Program:

```
let p1: string = "abc123";
let p2: string = "abc123";
if (p1 === p2)
   console.log("Password Matched");
else
   console.log("Password Mismatch");
```

Experiment 15: Maintenance Mode

Aim:

To check website maintenance mode.

Input:

true

Output:

Site Under Maintenance

Program:

```
let maintenance: boolean = true;


if (maintenance)
   console.log("Site Under Maintenance");
```

Experiment 16: Device Type Detection

Aim:

To detect device type.

Input:

mobile

Output:

Mobile View

Program:

```
let device: string = "mobile";


if (device === "mobile")
   console.log("Mobile View");
else
   console.log("Desktop View");
```

Experiment 17: Auto Logout Check

Aim:

To logout user after inactivity.

Input:

15

Output:

Auto Logout

Program:

```
let inactive: number = 15;


if (inactive >= 10)
   console.log("Auto Logout");
```

Experiment 18: Button Click Counter

Aim:

To count button clicks.

Input:

2 clicks

Output:

Clicked: 2

Program:

```
let clicks: number = 0;


clicks++;
clicks++;


console.log("Clicked:", clicks);
```

Experiment 19: Username Minimum Length

Aim:

To check minimum username length.

Input:

abc

Output:

Username Too Short

Program:

```
let user: string = "abc";
```

```
if (user.length < 4)
   console.log("Username Too Short");
else
   console.log("Valid Username");
```

Experiment 20: Gmail Domain Check

Aim:

To verify Gmail email ID.

Input:

user@gmail.com

Output:

Gmail User

Program:

```
let email: string = "user@gmail.com";


if (email.endsWith("@gmail.com"))
   console.log("Gmail User");
else
   console.log("Other Email");
```

Experiment 21: Remove Duplicate Elements from Array

Aim:

To remove duplicate elements from an array.

Input:

[1, 2, 2, 3, 4, 4]

Output:

[1, 2, 3, 4]

Program:

```
let arr: number[] = [1, 2, 2, 3, 4, 4];
let unique = Array.from(new Set(arr));
console.log(unique);
```

Experiment 22: Find Second Largest Element

Aim:

To find the second largest number in an array.

Input:

[10, 40, 30, 20]

Output:

30

Program:

```
let arr: number[] = [10, 40, 30, 20];
let sorted = arr.sort((a, b) => b - a);
console.log(sorted[1]);
```

Experiment 23: Count Occurrence of Characters

Aim:

To count frequency of each character in a string.

Input:

hello

Output:

h:1 e:1 l:2 o:1

Program:

```
let str: string = "hello";
let freq: any = {};
for (let ch of str) {
    freq[ch] = (freq[ch] || 0) + 1;
}
```

```
console.log(freq);
```

Experiment 24: Check Object Property Exists

Aim:

To check whether a property exists in an object.

Input:

{name:"Ravi", age:20}

property: age

Output:

Property Exists

Program:

```
let user = { name: "Ravi", age: 20 };
if ("age" in user)
   console.log("Property Exists");
else
   console.log("Property Not Found");
```

Experiment 25: Deep Copy vs Shallow Copy

Aim:

To demonstrate deep copy using spread operator.

Input:

{a:1, b:2}

Output:

Copied Object Created

Program:

```
let obj1 = { a: 1, b: 2 };
let obj2 = { ...obj1 };

console.log("Copied Object Created");
```

Experiment 26: Optional Parameter in Function

Aim:

To demonstrate optional parameters in TypeScript.

Input:

add(10, 20)

Output:

30

Program:

```
function add(a: number, b: number, c?: number): number {
   return c ? a + b + c : a + b;
}

console.log(add(10, 20));
```

Experiment 27: Default Parameter Function

Aim:

To demonstrate default parameter values.

Input:

greet()

Output:

Hello User

Program:

```
function greet(name: string = "User"): void {
   console.log("Hello " + name);
}
greet();
```

Experiment 28: Promise Example

Aim:

To demonstrate Promise usage in TypeScript.

Input:

No input

Output:

Promise Resolved

Program:

```
let promise = new Promise((resolve) => {
   resolve("Promise Resolved");
});

promise.then(result => console.log(result));
```

Experiment 29: Async and Await Example

Aim:

To demonstrate async and await.

Input:

No input

Output:

Data Loaded

Program:

```
async function loadData() {
   return "Data Loaded";
}

loadData().then(data => console.log(data));
```

Experiment 30: Error Handling with Custom Message

Aim:

To handle runtime error using try-catch.

Input:

10 / 0

Output:

Error Occurred

Program:

```
try {
   throw new Error("Error Occurred");
} catch (e: any) {
   console.log(e.message);
}
```

Experiment 31: Find Common Elements Between Two Arrays

Aim:

To find common elements between two arrays.

Input:

[1, 2, 3, 4]

[3, 4, 5, 6]

Output:

[3, 4]

Program:

```
let a: number[] = [1, 2, 3, 4];
let b: number[] = [3, 4, 5, 6];

let common = a.filter(x => b.includes(x));
console.log(common);
```

Experiment 32: Convert Array of Strings to Uppercase

Aim:

To convert all array elements to uppercase.

Input:

["apple", "banana"]

Output:

["APPLE", "BANANA"]

Program:

```
let fruits: string[] = ["apple", "banana"];


let result = fruits.map(f => f.toUpperCase());
console.log(result);
```

Experiment 33: Filter Even Numbers from Array

Aim:

To filter even numbers from an array.

Input:

[1, 2, 3, 4, 5, 6]

Output:

[2, 4, 6]

Program:

```
let nums: number[] = [1, 2, 3, 4, 5, 6];


let even = nums.filter(n => n % 2 === 0);
console.log(even);
```

Experiment 34: Sum of Array Elements using Reduce

Aim:

To calculate sum of array elements using reduce.

Input:

[10, 20, 30]

Output:

60

Program:

```
let arr: number[] = [10, 20, 30];
```

```
let sum = arr.reduce((total, val) => total + val, 0);
console.log(sum);
```

Experiment 35: Sort Objects by Property

Aim:

To sort array of objects based on age.

Input:

[{name:"A",age:25},{name:"B",age:20}]

Output:

Sorted by age

Program:

```
let users = [
    { name: "A", age: 25 },
    { name: "B", age: 20 }
];
users.sort((x, y) => x.age - y.age);
console.log(users);
```

Experiment 36: Find Maximum Value using Math.max

Aim:

To find maximum value in an array.

Input:

[5, 10, 20, 8]

Output:

20

Program:

```
let nums: number[] = [5, 10, 20, 8];


console.log(Math.max(...nums));
```

Experiment 37: Convert Object Keys to Array

Aim:

To convert object keys into an array.

Input:

{name:"Ravi", age:20}

Output:

["name", "age"]

Program:

```
let obj = { name: "Ravi", age: 20 };


let keys = Object.keys(obj);
console.log(keys);
```

Experiment 38: Convert Object Values to Array

Aim:

To convert object values into an array.

Input:

{name:"Ravi", age:20}

Output:

["Ravi", 20]

Program:

```
let obj = { name: "Ravi", age: 20 };
let values = Object.values(obj);
console.log(values);
```

Experiment 39: Closure Example

Aim:

To demonstrate closure in TypeScript.

Input:

No input

Output:

Count: 1

Count: 2

Program:

```
function counter() {
   let count = 0;
   return function () {
      count++;
      console.log("Count:", count);
   };
}

let inc = counter();
inc();
inc();
```

Experiment 40: Readonly Property

Aim:

To demonstrate readonly property in TypeScript.

Input:

No input

Output:

ID: 101

Program:

```
class User {
   readonly id: number = 101;
}
let u = new User();
console.log("ID:", u.id);
```

Experiment 41: Check Array is Empty or Not

Aim:

To check whether an array is empty.

Input:

[]

Output:

Array is Empty

Program:

```
let arr: number[] = [];


if (arr.length === 0)
   console.log("Array is Empty");
else
   console.log("Array is Not Empty");
```

Experiment 42: Merge Two Arrays

Aim:

To merge two arrays into one array.

Input:

[1,2]

[3,4]

Output:

[1, 2, 3, 4]

Program:

```
let a: number[] = [1, 2];
let b: number[] = [3, 4];


let merged = [...a, ...b];
console.log(merged);
```

Experiment 43: Remove Specific Element from Array

Aim:

To remove a specific element from an array.

Input:

[10,20,30]

20

Output:

[10, 30]

Program:

```
let arr: number[] = [10, 20, 30];
let remove = 20;
let result = arr.filter(x => x !== remove);
console.log(result);
```

Experiment 44: Check String Starts With Specific Word

Aim:

To check whether a string starts with a given word.

Input:

TypeScript Language

TypeScript

Output:

Starts With Word

Program:

```
let text: string = "TypeScript Language";
if (text.startsWith("TypeScript"))
   console.log("Starts With Word");
else
   console.log("Does Not Start");
```

Experiment 45: Convert Number Array to String Array

Aim:

To convert number array to string array.

Input:

[1,2,3]

Output:

["1","2","3"]

Program:

```
let nums: number[] = [1, 2, 3];


let strArr = nums.map(n => n.toString());

console.log(strArr);
```

Experiment 46: Find Index of an Element

Aim:

To find the index of an element in an array.

Input:

[10,20,30]

30

Output:

2

Program:

```
let arr: number[] = [10, 20, 30];

console.log(arr.indexOf(30));
```

Experiment 47: Convert JSON String to Object

Aim:

To convert JSON string into an object.

Input:

{"name":"Ravi","age":20}

Output:

Ravi 20

Program:

```
let jsonStr: string = '{"name":"Ravi","age":20}';
let obj = JSON.parse(jsonStr);
console.log(obj.name, obj.age);
```

Experiment 48: Convert Object to JSON String

Aim:

To convert object into JSON string.

Input:

{name:"Ravi",age:20}

Output:

{"name":"Ravi","age":20}

Program:

```
let obj = { name: "Ravi", age: 20 };
let json = JSON.stringify(obj);
console.log(json);
```

Experiment 49: Optional Chaining Operator

Aim:

To safely access object property using optional chaining.

Input:

{user:{name:"Ravi"}}

Output:

Ravi

Program:

```
let data: any = { user: { name: "Ravi" } };
console.log(data.user?.name);
```

Experiment 50: Nullish Coalescing Operator

Aim:

To assign default value using nullish coalescing operator.

Input:

null

Output:

Guest

Program:

```
let username: string | null = null;


let name = username ?? "Guest";
console.log(name);
```