**Typescript assignment 1**

**Exercise 1: Define Typed Arrays**

Define the following arrays with proper types and initialize them with values.

An array of numbers. Const Num: number [ ] = [1,2,3,4,5,6];

An array of strings. Const Strings: String [] = [“hello”, “World”];

An array of Booleans. Const Booleans: Boolean [] = [true, false];

A mixed array containing numbers and strings. Const mix: (numbers | Strings)[] = [1, “hello”,2,”world”];

**Exercise 2: Array Manipulation**

Perform the following operations on the given array:

Add "grapes" at the end.

Remove the first element.

Insert "pineapple" at index 2.

Reverse the array.

Let fruits: string [] = ["apple", "banana", "mango"];

fruits. Push(“grapes”);

console.log(fruits);

fruits. shift();

console.log(fruits);

fruits. splice(2,0, “pineapple”);

console.log(fruits);

fruits.reverse();

console.log(fruits);

**Exercise 3: Map, Filter, and Reduce**

Using the given array, perform the following:

Create a new array that doubles each number.

Filter out numbers greater than 50.

Find the sum of all numbers.

let numbers: number[] = [10, 20, 30, 40, 50, 60];

let doubled: number[] = numbers.map(num => num \* 2);

console.log("Doubled Numbers:", doubled);

let filtered: number[] = doubled.filter(num => num < 50);

console.log("Filtered Numbers (≤50):", filtered);

let sum: number = filtered.reduce((acc, num) => acc + num, 0);

console.log("sum of filtered num:", sum);

**Exercise 4: Find and FindIndex**

Using the given array:

Find the first number greater than 25.

Get the index of the number 40.

let values: number[] = [10, 20, 30, 40, 50];

let result:number[] = values.find(num=>num>25);

console.log("First number > 25:", result);

let index:number[] = result.indexof(40);

console.log("Index of 40:", index);

**Exercise 5: Sorting an Array**

Sort the following array in ascending and descending order.

let scores: number[] = [55, 23, 87, 12, 99, 38];

let ascending = scores.sort((a, b) => a - b);

console.log("Ascending order:", ascending);

let descending: number[] = scores.sort((a, b) => b – a);

console.log("Descending order:", descending);

**Exercise 6: Tuple Types**

Create a tuple that holds the following information about a student:

Name (string)

Age (number)

Enrolled (boolean)

Then, update the age of the student.

let student: [string, number, boolean] = ["John", 21, true];

student[1] = 21;

console.log(student);

**Exercise 7: Working with includes() and some()**

Given an array of roles, check:

If "Admin" exists in the array.

If at least one role starts with "Super".

let roles: string[] = ["User", "Editor", "Admin", "Moderator"];

let hasAdmin: boolean = roles.includes("Admin");

let hasSuper: boolean = roles.some(role => role.startsWith("Super"));

console.log("Contains Admin:", hasAdmin);

console.log("Has Super role:", hasSuper);

**Exercise 8: Flattening an Array**

Flatten the following nested array using the flat() method.

let nestedArray: (number | number[])[] = [1, [2, 3], [4, [5, 6]]];

let flattenedArray = nestedArray.flat(Infinity);

console.log(flattenedArray);

**Exercise 9: Merging and Removing Duplicates**

Merge the two arrays and remove duplicates.

let arr1: number[] = [1, 2, 3, 4];

let arr2: number[] = [3, 4, 5, 6];

let uniqueArray: number[] = [...new Set([...arr1, ...arr2])];

console.log(uniqueArray);

**Exercise 10: Interface with Arrays**

Define an interface Employee with the following properties:

name (string)

age (number)

position (string)

Then, create an array of employees and filter those older than 30.

interface Employee {

name: string;

age: number;

position: string;

}

let employees: Employee[] = [

{ name: "jp", age: 28, position: "Developer" },

{ name: "rs", age: 35, position: "Manager" },

{ name: "ns", age: 40, position: "tester" },

{ name: "fz", age: 25, position: "Interns" }

];

let filteredEmployees: Employee[] = employees.filter(emp => emp.age > 30);

console.log(filteredEmployees);