Typescript Question Paper

* Write a TypeScript program that declares a variable using let inside a block and const outside a block. Show the difference in scope by trying to access them outside their blocks.
* Create a const array of numbers. Try reassigning the array to another array and also push a new number into the array. Explain the difference in behaviour.
* Declare a strongly typed array of strings that stores names of cities. Write a function that accepts this array and prints all cities in uppercase.
* Create a strongly typed array of numbers and write a function that returns the sum of only even numbers in the array.
* Write two variables, one using type inference and one using explicit type annotation, both storing a string "Hello Bosch Employees!". Print their types at runtime.
* Create a function that takes a number as an argument and returns its square. First, implement it without type annotations, then with explicit type annotations. Show how TypeScript reacts to invalid input in both cases.
* Convert the following function into an arrow function:

function greet(name: string): string {

return "Hello " + name;

}

* Write an arrow function that filters numbers greater than 50 from an array of numbers. Test it with [10, 55, 32, 75, 90].
* Create an enum Days for all 7 days of the week. Write a function that takes a Days enum value and prints if it’s a weekday or weekend.
* Define an enum Direction { North, South, East, West }. Write a function that takes a Direction value and prints corresponding travel message.
* Write a TypeScript program using for-of to iterate over an array of numbers and print the square of each number.
* Create a map of country codes and country names (e.g., { IN: "India", US: "United States" }). Use for-of with Object.entries() to print both code and country.
* Define an abstract class Shape with an abstract method area(). Implement Rectangle and Circle classes that extend Shape and calculate area.
* Create an abstract class Employee with properties name and salary. Extend it in a concrete class Manager and add a method to calculate bonus.
* Create an interface Person with properties name, age, and a method introduce(). Implement it in a class Student.
* Write an interface Vehicle with method startEngine(). Implement two classes Car and Bike that implement the interface.
* Create a generic class Storage<T> with methods addItem(item: T) and getAllItems(). Demonstrate with string and number data types.
* Write a generic class KeyValue<K, V> to store key-value pairs. Add a method to print all pairs.
* Create a class decorator @Logger that logs the creation of a class instance. Apply it to a class Product.
* Write a method decorator @LogExecutionTime that logs how long a function takes to execute. Apply it to a method that sorts an array of numbers.