TypeScript Practice Sheet

1. Basic Types

1. Define a variable for a user's age, ensuring that it cannot hold non-numeric values. Write a function that takes this age as input and determines if the user is eligible to vote.
2. Create a variable to store the name of a product. Ensure it can only hold string values. Write a function that capitalizes the product name.
3. Create a variable to store a boolean flag for "isAdmin" and write a function that returns a message depending on whether the user is an admin or not.
4. Write a function that accepts a variable of type string | number and returns its length if it is a string, or its square if it is a number.
5. Define a variable for the current date and write a function that checks if this date is a weekend or a weekday.

---

2. Array and Objects

1. Create an array of product prices. Write a function that calculates the total price of all products in the array.
2. Define an object to represent a user profile with fields: name, email, and age. Write a function that accepts this object and prints a user-friendly summary.
3. Write a function that takes an array of user objects (each with id and name properties) and returns an array of names.
4. Define a tuple to store latitude and longitude coordinates. Write a function that takes the tuple and returns a readable string format of the location.
5. Create an array of strings representing programming languages. Write a function to check if "TypeScript" exists in the array.

---

3. Functions

1. Write a function that takes two numbers and returns their sum. Ensure the parameters and return type are strictly typed.
2. Write a function that takes a callback function as a parameter. The callback should accept a string and return a number. Demonstrate its use.
3. Create a function that accepts an object with optional properties firstName and lastName. Return the full name if both are provided; otherwise, return "Unknown".
4. Write a function that takes a rest parameter of numbers and returns their average.
5. Implement a function that accepts an array of product objects (each with id and price) and a callback function. The callback should filter products based on the price and return a new array.

---

4. Assertions

1. Write a function that takes a string input and asserts that it is not null or undefined before returning its uppercase value.
2. Define a variable as unknown and use a type assertion to safely assign its value to a string variable.
3. Write a function that accepts a DOM element ID and asserts that the element exists before returning it.
4. Write a function that takes a JSON string and uses type assertions to parse it into a specific object type.
5. Create a function that accepts a union type (string | number) and uses assertions to perform different operations depending on the type.

---

5. Classes

1. Create a User class with properties id, name, and email. Add a method greet that returns a greeting message with the user's name.
2. Create a Product class with properties id, name, and price. Add a method to calculate the price after applying a discount.
3. Write a class Car with properties make, model, and year. Add a method to check if the car is considered a classic (older than 20 years).
4. Create a class BankAccount with properties accountNumber, balance, and ownerName. Add methods to deposit and withdraw money.

5. Create a class Employee with properties id, name, and department. Add a static method to count the total number of employees created.

---

6. Index Signatures and Keyof Assertions

1. Create an object that maps user IDs (numbers) to user names (strings). Write a function to get the name by ID using an index signature.
2. Write a function that takes an object and a key (using keyof) and returns the value of that key in the object.
3. Create an object with dynamic properties where the keys are strings and the values are boolean flags. Write a function that returns all keys with a true value.
4. Write a function that takes a record of string keys and number values, and calculates the total of all the values.
5. Define a type for an object that has keys as string product names and values as their stock numbers. Write a function to reduce the stock of a product by a given amount.

---

7. Generics

1. Write a generic function that accepts an array of any type and returns the first element.
2. Create a generic class Box<T> that has a value property and a method to set and get the value.
3. Write a generic function that merges two objects into one. Ensure type safety.
4. Create a generic interface Response<T> with properties status and data. Write a function that accepts this response and logs the data if the status is "success".
5. Implement a generic function to find the maximum value in an array of numbers or strings.

---

8. Utility Types

1. Define a User type with id, name, and email. Use Pick to create a type that only includes name and email.

2. Create a type for a Task with id, title, and isCompleted. Use Omit to exclude the id property.
3. Write a function that takes a Partial<User> and updates the user profile with default values for missing fields.
4. Use Readonly to create a type for a constant object and write a function to display its contents.
5. Create a type Product with name and price, and use Record to create a type for a product catalog with product names as keys.

---

9. Mixed

1. Combine Basic Types and Generics: Write a generic function that accepts an array of user names (strings) and returns an array of user IDs (numbers).
2. Combine Classes and Functions: Create a Task class with id and title properties. Add a method to mark the task as completed and another function to display all tasks.
3. Combine Utility Types and Index Signatures: Define a UserPermissions type using Record<string, boolean> and write a function to check if a user has a specific permission.
4. Combine Assertions and Generics: Write a function that takes an unknown input and asserts it as an array of a specific type using generics.
5. Combine Array and Objects with Utility Types: Create an array of Readonly objects, each representing a book with title and author. Write a function to display all book details.