1. Declare variables for the following:
    - A string to store a person's name.
    - A number to store their age.
    - A boolean to indicate if they are a student.

2. Write a TypeScript function addNumbers(a: number, b: number): number that takes two numbers as parameters and returns their sum.

3. Create an array of strings to store the names of five cities. Write a function to print each city in the array.

4. Use a tuple to store a product's details: `id (number)`, `name (string)`, and `price (number)`. Write a function that takes the tuple and prints the product details.

5. Write a function `greet` that returns a greeting message like "Hello, John!".

6. Write a function `calculateTotal` where `discount` is optional. If no discount is provided, it should assume a default value of 0%.

100 : 5%=>95
100=>100

7. Write a function `printAddress(street: string, city: string, country?: string): string`. If `country` is not provided, return a string like `"Street, City"`. Otherwise, include the country in the output.

8. Define an interface User with properties: username (required) email (optional) phoneNumber (optional) Write a function that takes a User object and prints their details. If email or phoneNumber is missing, handle it gracefully.

9. Create a class `Product` with properties: `id`, `name`, `price`, and `quantity`.
- Add methods to:
    - Update the quantity.
    - Calculate the total price for the available stock.

10. Write a function to create a list of `Product` objects and demonstrate the methods.

11. Define an interface `Car` with properties `brand`, `model`, and `year`. Write a function that accepts a `Car` object and returns a string summarizing the car's details.

12. Define a `Product` with properties like `id`, `name`, `price`, and `quantity`. Write a function that calculates the total price of all products in a cart.

13. Create a function `findMax(numbers: number[]): number` that takes an array of numbers and returns the largest number.

14. Create a simple `Person` class with properties `firstName` and `lastName` and a method `getFullName()` that returns the full name.

15. Define an enum `Color` with values `Red`, `Green`, and `Blue`. Write a function that takes a `Color` and returns its string representation.

16. Create an interface `Student` with `name`, `age`, and `grade`. Write a function to calculate the average grade of an array of `Student` objects.

17. **Inventory Management**:
    Create a class `Product` with properties: `id`, `name`, `price`, and `quantity`.

- Add methods to:

    - Update the quantity.
    - Calculate the total price for the available stock.
- Write a function to create a list of `Product` objects and demonstrate the methods.

18. **Library Management**:

- Create a `Library` class with properties: `name` and an array of `Book` objects (use an interface for `Book` with properties like `title`, `author`, `isAvailable`).
- Add methods to:
    - Add a new book to the library.
    - Lend a book (update its `isAvailable` property to `false`).
    - Return a book (update `isAvailable` to `true`).

19. Write a generic function `findUnique<T>(arr: T[]): T[]` that takes an array of any type and returns a new array with only the unique elements.

20. Create a function `truncateString(str: string, maxLength: number): string` that truncates the string to the specified length and adds "..." at the end if the string exceeds the length.
    Example:

- Input: `truncateString("TypeScript is awesome!", 10)`
- Output: `"TypeScript..."`

21. Employee Management:
    Define an interface Employee with properties id, name, department, and position.

    Write a function to filter employees based on their department.

    Use Partial<Employee> to allow updating only specific properties of an employee.

22. Define an enum `TrafficLight` with values `Red`, `Yellow`, and `Green`.

- Write a function `getAction(light: TrafficLight): string` that returns:
    - "Stop" for `Red`.
    - "Get Ready" for `Yellow`.
    - "Go" for `Green`.