Object Assignments

Learning Objectives:

In this assignment, you'll gain practical experience with TypeScript type aliases, a powerful feature for defining object structures. By working with employee data, car details, and colorful T-shirts, you'll solidify your understanding of:

- **Type Aliases:** Creating custom types to represent complex objects with specific properties.
- **Nested Objects:** Organizing data within objects using nested structures for hierarchical relationships.
- **Union Types:** Restricting a property's value to a set of allowed options (e.g., "Manager", "Engineer", "Intern").
- **Optional Properties:** Allowing for objects with missing properties without causing errors.
- **Code Clarity:** Employing descriptive variable and function names, comments, and well-structured code to enhance readability and maintainability.

The Assignment:

This assignment is divided into three parts, each focusing on using type aliases to model real-world scenarios:

Part 1: Employee Data

Challenge:

1. Design a type alias named `Employee` to represent an employee object with properties like `name` (string), `department` (string), and `role` (string).
2. Include an optional nested object named `contact` to hold phone and email information (if provided).
3. Employ a union type for the `role` property to restrict it to "Manager", "Engineer", or "Intern".
4. Make the `skills` property an optional array of strings for listing an employee's skills (if any).

Part 2: Car Details

Challenge:

1. Design a type alias named `Car` to represent a car object.
2. Include a nested object named `engine` within `Car`, containing a property named `horsepower` (number).
3. Define a function named `getHorsepower` directly within the `Car` type alias to retrieve the engine's horsepower.

Part 3: Colorful T-Shirts

Challenge:

1. Design a type alias named `Product` to represent a T-shirt object with properties like `name` (string), `price` (number), and `color` (string).
2. Include a nested object named `inventory` within `Product`. This `inventory` object should have two properties:
    - `stock` (number): Represents the number of T-shirts available.
    - `colorOptions` (optional array of strings): Lists available colors (if any).
3. Inside the `inventory` object, define a function named `changeColor`. This function accepts a new color string as an argument. When called, it should:
    - Update the `color` property of the `Product` object.
    - Adjust the `price` based on the new color (implement your own logic, e.g., increase by 10% for red, decrease by 5% for blue).