Part 1: Dart Types

1. Built-in Types

- **Task**: Write a Dart program that demonstrates the usage of various built-in types such as `int`, `double`, `String`, `bool`, and `List`. Create variables of each type and perform basic operations on them.
  Example:
  - Create an integer variable to represent age.
  - Create a double variable to represent height.
  - Create a string variable to represent a person's name.
  - Create a boolean variable to represent if the person is a student.
  - Create a list to store the person's grades.

2. Records

- **Task**: Implement a Dart record to group related data together. Create a record for a `Book` that includes the title, author, and number of pages. Write a function to display the record's contents.
  Example:
  - Define a record `(String title, String author, int pages)` for a book.
  - Use pattern matching to extract and display the book's details.

3. Collections

- **Task**: Write a Dart program that demonstrates the usage of collections like `List`, `Set`, and `Map`. Create a shopping list using `List`, a collection of unique product categories using `Set`, and a map to store product prices.
  Example:
  - Create a list to hold items in a shopping cart.
  - Create a set to hold unique product categories (e.g., electronics, clothing).
  - Create a map to store product names as keys and prices as values.

4. Generics

- **Task**: Implement a function using generics in Dart to create a collection of any type. Write a generic class `Box<T>` that holds an item of type `T`. Demonstrate storing items of different types (e.g., int, String) in the box.
  Example:
  - Create a class `Box<T>` with a field `item` of type `T`.
  - Instantiate the `Box` class for different data types (e.g., integers, strings).
  - Write a method that returns the item stored in the box.

5. Typedefs

- **Task**: Write a Dart program that demonstrates the usage of `typedef` to define a function signature. Create a typedef for a function that takes two integers and returns their sum.
  Example:
  - Define a typedef `MathOperation` for a function that takes two `int` values and returns an `int`.
  - Implement two functions, `add` and `multiply`, that match the `MathOperation` typedef.
  - Use the typedef in a function to apply the operation to given values.

6. Type System

- **Task**: Explain Dart's sound type system and null safety. Write a program that demonstrates the use of nullable and non-nullable types, including how to handle possible null values safely.
  Example:
  - Define a non-nullable variable and initialize it with a value.
  - Define a nullable variable using `?` and assign `null` to it.
  - Demonstrate using `if` checks and the null-aware operator (`?.`) to safely access nullable variables.

Part 2: Dart Patterns

1. Overview & Usage

- **Task**: Write a short overview of patterns in Dart and their common usage. Include examples of destructuring patterns, switch statements with patterns, and how Dart supports pattern matching with data structures.
  Example:
  - Explain what patterns are in Dart.
  - Demonstrate using a pattern in a `switch` statement to match against a list of values.

2. Pattern Types

- **Task**: Demonstrate different pattern types in Dart, such as:
  - **List patterns**: Decompose a list using patterns.
  - **Object patterns**: Match properties of an object.
- *Example*:
  - Define a list pattern to extract the first and second elements of a list.
  - Define an object pattern to extract specific fields from a class.

3. Applied Tutorial

- **Task**: Write a tutorial that applies patterns to a real-world example. Create a program that takes a user's input (name and age) and matches it against predefined patterns to display a custom message.
  Example:
    - Use patterns to check the age and display if the user is a minor or an adult.
    - Match the input with a specific name pattern and greet the user accordingly.

Submission Instructions:

- Submit the Dart code files for each task.
- Ensure proper use of comments to explain the code logic.
- Provide the output of your programs where applicable.
- Upload on github