Assignment: Functions in Dart

Task 1: Basic Function Definition

1. Write a function `greetUser()` that prints `"Hello, User!"`.
2. Call this function in `main()`.

Expected Output:

Hello, User!

Task 2: Function with Return Type

1. Create a function `square(int number)` that returns the square of the number.
2. Call the function and print the returned value.

Example Call:

int result = square(4);
print(result);

Expected Output:

16

Task 3: Positional and Named Parameters

Write a function `introducePerson(String name, int age, {String? city})` that prints:
My name is [name], I am [age] years old, and I live in [city].

1.
2. The `city` parameter should be **optional**. If not provided, print `"City not specified"`.

Example Call:

```
introducePerson("Ali", 25, city: "Lahore");
introducePerson("Sara", 30);
```

Expected Output:

```
My name is Ali, I am 25 years old, and I live in Lahore.
My name is Sara, I am 30 years old, and I live in City not specified.
```

Task 4: Required Named Parameters

1. Write a function `calculateBMI({required double weight, required double height})` that returns **BMI = weight / (height * height)**.
2. Call this function with named arguments.

Example Call:

```
double bmi = calculateBMI(weight: 70, height: 1.75);
print(bmi);
```

Task 5: Arrow Function

1. Convert the following function into an arrow function:

```
int multiply(int a, int b) {
  return a * b;
}
```

Example Call:

```
int result = multiply(3, 5);
print(result);
```

Expected Output:

```
15
```

Task 6: Function as a Parameter

1. Write a function `applyOperation(int a, int b, Function operation)` that takes two integers and a function as a parameter.
2. Pass addition and multiplication functions to `applyOperation()`.

Example Call:

```
int add(int x, int y) => x + y;
int multiply(int x, int y) => x * y;

applyOperation(5, 3, add);
applyOperation(5, 3, multiply);
```

Expected Output:

```
Result: 8
Result: 15
```

Task 7: Function Returning a Function

1. Write a function `chooseOperation(String opType)` that returns a function based on `opType`.
2. If `opType` is `"add"`, return an addition function.
3. If `opType` is `"multiply"`, return a multiplication function.

Example Call:

```
var operation = chooseOperation("add");
print(operation(4, 6)); // Output: 10
```

Task 8: Function with List Processing

1. Create a function `filterEvenNumbers(List<int> numbers)` that returns a list containing only even numbers.
2. Use `where()` function inside it.

Example Call:

```
List<int> evens = filterEvenNumbers([1, 2, 3, 4, 5, 6]);
print(evens);
```

Expected Output:

```
[2, 4, 6]
```

Task 9: Function with Map Processing

1. Given a list of maps representing students with their scores:

```
List<Map<String, dynamic>> students = [
  {"name": "Ali", "score": 85},
  {"name": "Sara", "score": 90},
  {"name": "John", "score": 78}
];
```

2. Write a function `printTopStudent(List<Map<String, dynamic>> students)` that finds and prints the student with the highest score.

Expected Output:

```
Top student: Sara with score 90
```