Lab 2: Classes, objects, functions and app backend practice

Scenario:
You are developing a feature for a popular Pakistani e-learning platform called **"Seekho Pakistan"**. Your task is to model the system for managing courses, instructors, and student enrollments using Dart.

Part 1: Creating the Core Classes

Create the following classes with their respective properties and methods.

1. Class: Instructor

Properties:

- name (String): The name of the instructor. (e.g., "Dr. Saima Ali")
- bio (String): A short biography. (e.g., "Data Scientist and AI researcher from NUST.")
- expertise (List<String>): A list of their areas of expertise. (e.g., ['Python', 'Machine Learning', 'Data Analysis'])

Constructor: Create a constructor that requires name and bio. expertise should be an optional parameter, defaulting to an empty list.

Method:

- addExpertise(String newSkill): A method to add a new skill to the instructor's expertise list.
- displayProfile(): A method that prints the instructor's profile neatly.

  Example output: "Instructor: Dr. Saima Ali\nBio: Data Scientist...\nExpertise: Python, Machine Learning, Data Analysis"

2. Class: Course

Properties:

- title (String): The title of the course. (e.g., "Web Development with Flask")

- instructor (Instructor): The instructor of the course (an instance of the Instructor class).
- price (double): The price in Pakistani Rupees (PKR).
- duration (String): The total duration. (e.g., "8 weeks")
- isPublished (bool): Whether the course is published and available for enrollment.

Constructor: Create a constructor that requires all these properties.

Method:

- publishCourse(): A method that sets isPublished to true and prints a message that the course is now live.
- getCourseInfo(): A method that returns a string with the course's key information.

3. **Class:** StudentEnrollment

Properties:

- studentName (String): The name of the student. (e.g., "Bilal Raza")
- course (Course): The course the student is enrolled in.
- enrollmentDate (DateTime): The date of enrollment.
- progress (double): The percentage of course completion (0 to 100).

Constructor: Create a constructor that requires studentName, course, and enrollmentDate. progress should be an optional parameter, defaulting to 0.0.

Method:

- updateProgress(double newProgress): A method to update the progress. It should validate that newProgress is between 0 and 100.
- generateCertificate(): A method that checks if progress is 100%. If it is, it prints a certificate message (e.g., "Certificate of Completion awarded to Bilal Raza for Web Development with Flask!"). If not, it prints a message encouraging the student to continue.

Part 2: Implementing Various Functions

1. Function with Optional Positional Parameters: calculateTotalCost
Create a function that calculates the total cost for a student.

Parameters:

- basePrice (double): Required. The price of the course.
- [discountVoucher] (double): Optional. A discount voucher amount in PKR.
- [hasScholarship] (bool): Optional. If true, apply a 50% scholarship after the voucher.

Returns: (double) The final total cost.

Example Call:

```
double cost = calculateTotalCost(5000); // returns 5000
double cost2 = calculateTotalCost(5000, 1000); // returns
4000
double cost3 = calculateTotalCost(5000, 1000, true); //
returns 2000 -> (5000 - 1000 = 4000) * 0.5 = 2000
```

2. Function with Named Parameters and Default Values:
sendNotification
Create a function to simulate sending a notification to a user.

Parameters:

- message (String): Required. The notification content.
- {String type = 'Email'}: Optional named parameter. The type of notification (e.g., 'Email', 'SMS', 'Push').
- {bool isUrgent = false}: Optional named parameter. If true, prepend "URGENT: " to the message.

Returns: (void) This function should just print the formatted notification.

Example Call:

```
sendNotification("Your class starts in 10 mins!");
// Output: [Email] Notification: Your class starts in 10
mins!

sendNotification("Password changed successfully", type:
'SMS');
// Output: [SMS] Notification: Password changed
successfully
```

```
sendNotification("Server maintenance tonight", type:
'Push', isUrgent: true);
// Output: [Push] Notification: URGENT: Server
maintenance tonight
```

3. Higher-Order Function: processEnrollments

Create a function that processes a list of enrollments and performs a specific action on each one. This is useful for batch operations like sending welcome emails or checking progress.

Parameters:

- List<StudentEnrollment> enrollments: The list of enrollments to process.
- void Function(StudentEnrollment) action: A function that defines what to do with each enrollment.

Behavior: This function should use a for-in loop to apply the action function to every enrollment in the list.

Example Call:

```
// Check who is eligible for a certificate
processEnrollments(allEnrollments, (enrollment) {
  if (enrollment.progress == 100) {
    enrollment.generateCertificate();
  }
});

// Send a notification to all students in a list
processEnrollments(allEnrollments, (e) =>
sendNotification("New lecture uploaded!", type: 'Push'));
```

Part 3: Putting It All Together (Main Function)

In your main() function, perform the following steps:

1. **Create Instances:**

   Create an Instructor object for "Sir Usman Ahmed", a "Mobile App Developer with 10 years of experience". Add expertise in ['Flutter', 'Dart', 'Firebase'].

   Create a Course object titled "Flutter Zero to Hero" taught by the instructor you just created. Set the price to 3500 PKR, duration to "6 weeks", and initially set it as not published.

2. **Publish the Course:**

   Use the publishCourse() method to publish the course.

3. **Enroll a Student:**

Create a StudentEnrollment object for a student named "Ayesha Siddiqa" in the Flutter course. Use DateTime.now() for the enrollment date.

4. **Use Your Functions:**

    Calculate the total cost for Ayesha if she uses a 500 PKR voucher and has a scholarship. Print the result.

    Send her an urgent SMS notification: "Welcome to Seekho Pakistan!".

5. **Simulate Learning Progress:**

    Update Ayesha's progress to 45%, then later to 100%.

    Try to generate a certificate after each progress update.

6. **Demonstrate Higher-Order Function:**

    Create a list of enrollments and add Ayesha's enrollment to it.

    Use the processEnrollments function to send a push notification to all students in the list about a "Live Q&A Session next week".