Sure 👍 here's the **full-length answer for Question 1(a): Install Flutter and Dart SDK**, written clearly as you'd write it in your **UID Lab record**.

1(a) Installation of Flutter and Dart SDK

Aim:

To install and set up Flutter and Dart SDK for developing cross-platform mobile applications.

Theory:

Flutter is an open-source UI framework developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Dart is the programming language used to build Flutter applications.

Before creating Flutter apps, we must install both Flutter SDK and Dart SDK, and configure the environment variables properly.

Steps to Install Flutter and Dart SDK:

Step 1: Download Flutter SDK

- Visit the official Flutter website: https://flutter.dev/docs/get-started/install
- Choose your operating system (Windows, macOS, or Linux).
- Download the Flutter SDK zip file.

Step 2: Extract the Flutter SDK

- Extract the downloaded zip file to a desired location on your system.
  Example:
    - Windows → `C:\src\flutter`
    - macOS/Linux → `/usr/local/flutter`

Step 3: Set Environment Variable (PATH)

- Add the Flutter `bin` folder to your system's PATH.
    - **Windows:**
        1. Go to *System Properties* → *Environment Variables*

2. Under *System Variables*, find **Path** and click *Edit*.
3. Add the path `C:\src\flutter\bin`

Linux/macOS:
Add the following line to your terminal profile file (`.bashrc`, `.zshrc`, etc.):

```
export PATH="$PATH:/usr/local/flutter/bin"
```

-

Step 4: Verify Installation
Open a terminal or command prompt and type:

```
flutter doctor
```

-
- Flutter will check for required dependencies and display their status.
- Ensure all checks show a green tick ✔.

Step 5: Install Required Tools

- Install the following tools if prompted by `flutter doctor`:
  - Android Studio (for Android SDK and Emulator)
  - Visual Studio Code (for code editing)
  - Git (for cloning Flutter projects)
  - Chrome (for web testing, optional)

Step 6: Install Dart SDK (if not included)

- Dart comes bundled with Flutter, but can also be installed separately from https://dart.dev/get-dart

Verify Dart installation:

```
dart --version
```

-

Step 7: Test Installation

Create a sample Flutter project:

```
 flutter create my_app
cd my_app
flutter run
```

-
- The app should launch successfully on your connected device or emulator.

Result:

Flutter and Dart SDK were successfully installed and configured on the system. The installation was verified using the `flutter doctor` command, and a sample Flutter application was executed successfully.

1(b) Design a Responsive UI that Adapts to Different Screen Sizes

Aim:

To design and implement a responsive Flutter UI that automatically adjusts its layout based on different screen sizes such as mobile, tablet, and desktop.

Program Code:

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

 Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

        appBar: AppBar(title: const Text('Responsive UI')),

        body: LayoutBuilder(

          builder: (context, constraints) {
```

```
            if (constraints.maxWidth < 600) {
              return const Center(child: Text('📱 Mobile View'));
            } else {
            return const Center(child: Text('💻 Tablet/Desktop
View'));
            }
          },
        ),
      ),
    );
  }
}
```

Output:

- When the app runs on a **mobile screen (width < 600 px)** → Displays **"📱 Mobile View"**.
- When the app runs on a **tablet/desktop (width ≥ 600 px)** → Displays **"💻 Tablet/Desktop View"**.
-

2(a) Explore Various Flutter Widgets

Aim:

To understand and use basic Flutter widgets like Text, Image, and Container.

Code:

```
import 'package:flutter/material.dart';

void main() => runApp(const MaterialApp(home: WidgetsDemo()));
```

```
class WidgetsDemo extends StatelessWidget {
  const WidgetsDemo({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Widgets Demo')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text('Hello Flutter!', style: TextStyle(fontSize:
24)),

Image.network('https://flutter.dev/images/flutter-logo-sharing.png',
width: 100),
            Container(
              padding: const EdgeInsets.all(10),
              color: Colors.blue,
              child: const Text('Inside Container', style:
TextStyle(color: Colors.white)),
            ),
          ],
        ),
      ),
    );
  }
```

```
}
```

Output:

Displays text, an image, and a colored container.

2(b) Fibonacci Series using Function

Aim:

To generate Fibonacci numbers using a function.

Code:

```
import 'dart:io';

void main() {
  stdout.write('Enter count: ');
  int n = int.parse(stdin.readLineSync()!);
  fib(n);
}


void fib(int n) {
  int a = 0, b = 1;
  for (int i = 0; i < n; i++) {
    stdout.write('$a ');
    int c = a + b;
    a = b;
    b = c;
  }
}
```

Output:

```
Enter count: 6

0 1 1 2 3 5
```

3(a) Implement Different Layout Structures using Row, Column, and Stack

Aim:

To design layouts using **Row**, **Column**, and **Stack** widgets in Flutter.

Code:

```
import 'package:flutter/material.dart';

void main() => runApp(const MaterialApp(home: LayoutDemo()));


class LayoutDemo extends StatelessWidget {
  const LayoutDemo({super.key});


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Layouts Demo')),
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
            children: const [
              Icon(Icons.star, color: Colors.red),
              Icon(Icons.star, color: Colors.green),
              Icon(Icons.star, color: Colors.blue),
            ],
          ),
          const SizedBox(height: 20),
          Stack(
            alignment: Alignment.center,
            children: [
              Container(width: 100, height: 100, color:
Colors.blue),
              Container(width: 60, height: 60, color:
Colors.yellow),
            ],
          ),
        ],
      ),
    );
  }
}
```

Output:

- **Row:** Shows 3 colored stars in a row.
- **Column:** Vertically arranges widgets.
- **Stack:** Overlaps containers.

3(b) Learn about Stateless Widgets

Aim:

To understand the concept of a **stateless widget** in Flutter.

Theory:

A **stateless widget** is a widget that does **not change its state** after it's built.
Its appearance remains constant unless rebuilt by external changes.

Code:

```
import 'package:flutter/material.dart';

void main() => runApp(const MaterialApp(home: MyWidget()));


class MyWidget extends StatelessWidget {

  const MyWidget({super.key});


  @override

  Widget build(BuildContext context) {

    return const Scaffold(

      body: Center(child: Text('This is a Stateless Widget')),

    );

  }

}
```

Output:

Displays text: *"This is a Stateless Widget"* at the center of the screen.

4(a) Set up Navigation Between Different Screens using Navigator

Aim:

To implement screen navigation in Flutter using the **Navigator** widget.

Code:

```
import 'package:flutter/material.dart';

void main() => runApp(const MaterialApp(home: FirstScreen()));


class FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Screen')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Second Screen'),
          onPressed: () {
            Navigator.push(context,
              MaterialPageRoute(builder: (context) => const
SecondScreen()));
          },
        ),
      ),
```

```
    );
  }
}

class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Back to First'),
          onPressed: () => Navigator.pop(context),
        ),
      ),
    );
  }
}
```

Output:

- Tap “Go to Second Screen” → navigates to the second page.
- Tap “Back to First” → returns to the first screen.

4(b) Check Palindrome String using Dart

Aim:

To write a program that checks if a string is a **palindrome** or not.

Code:

```
import 'dart:io';

void main() {
  stdout.write('Enter a string: ');
  String str = stdin.readLineSync()!;
  String rev = str.split('').reversed.join('');
  if (str == rev) {
    print('Palindrome');
  } else {
    print('Not Palindrome');
  }
}
```

Output:

```
Enter a string: madam

Palindrome
```

5(a) Implement Navigation with Named Routes

Aim:

To navigate between multiple screens using **named routes** in Flutter.

Code:

```dart
import 'package:flutter/material.dart';
void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const FirstScreen(),
        '/second': (context) => const SecondScreen(),
      },
    );
  }
}

class FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Screen')),
      body: Center(
```

```
        child: ElevatedButton(
          onPressed: () => Navigator.pushNamed(context, '/second'),
          child: const Text('Go to Second'),
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Back'),
        ),
      ),
    );
  }
}
```

Output:

- Click **“Go to Second”** → navigates to the second screen.
- Click **“Back”** → returns to the first screen.

5(b) Check Whether a Number is Even or Odd

Aim:

To check whether a given number is even or odd.

Code:

```
import 'dart:io';

void main() {
  stdout.write('Enter a number: ');
  int n = int.parse(stdin.readLineSync()!);
  if (n % 2 == 0) {
    print('Even Number');
  } else {
    print('Odd Number');
  }
}
```

Output:

```
Enter a number: 7

Odd Number
```

6(a) Apply Styling using Themes and Custom Styles

Aim:

To apply simple styling using themes in Flutter.

Code:

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());


class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primaryColor: Colors.blue,
        textTheme: const TextTheme(
          bodyMedium: TextStyle(fontSize: 22, color: Colors.red),
        ),
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text('Theme Example')),
        body: const Center(child: Text('Hello Flutter!')),
      ),
    );
  }
```

```
}
```

Output:

Shows a blue app bar and red text “Hello Flutter!” in the center.

6(b) Find All Divisors of a Number

Aim:

To display all divisors of a given number.

Code:

```
import 'dart:io';

void main() {
  stdout.write('Enter a number: ');
  int n = int.parse(stdin.readLineSync()!);
  stdout.write('Divisors: ');
  for (int i = 1; i <= n; i++) {
    if (n % i == 0) stdout.write('$i ');
  }
}
```

Output:

```
Enter a number: 12
Divisors: 1 2 3 4 6 12
```

7 A) Add animations to UI elements using Flutter's animation framework

Aim:

To implement a simple animation in Flutter using the `AnimatedContainer` widget.

Code:

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());


class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(home: AnimateDemo());
  }
}


class AnimateDemo extends StatefulWidget {
  const AnimateDemo({super.key});
  @override
  State<AnimateDemo> createState() => _AnimateDemoState();
}


class _AnimateDemoState extends State<AnimateDemo> {
  double size = 100;
  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Simple Animation')),
      body: Center(
        child: GestureDetector(
          onTap: () => setState(() => size = size == 100 ? 200 :
100),
          child: AnimatedContainer(
            duration: const Duration(seconds: 1),
            color: Colors.blue,
            width: size,
            height: size,
          ),
        ),
      ),
    );
  }
}
```

Output:

Displays a **blue square** in the center that **smoothly grows and shrinks** when tapped.

7 B) Create a program to ask the user to enter their name and age, and tell how many years until 100

Aim: To create a simple Dart program that calculates how many years are left for the user to turn 100.

Code:

```
import 'dart:io';
void main() {
  stdout.write("Enter your name: ");
  String name = stdin.readLineSync()!;
  stdout.write("Enter your age: ");
  int age = int.parse(stdin.readLineSync()!);
  int years = 100 - age;
  print("$name, you will be 100 years old in $years years!");
}
```

Output Example:

```
Enter your name: Bhargav
Enter your age: 20
Bhargav, you will be 100 years old in 80 years!
```