DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

**LAB SESSION 4**

**EXPLORING MAPS IN DART**

**Objective:**

The objective of this lab is to introduce students to the concept of maps in Dart and provide hands-on experience in working with key-value pairs. By the end of this lab, you will be familiar with creating, manipulating, and utilizing maps to solve various problems in Dart programming.

**Introduction:**

Maps are an essential data structure in Dart that allow you to store and manipulate key-value pairs. In Dart, maps are represented by the Map class, which provides a flexible and efficient way to organize data. Each key in a map is unique, and the keys can be of any data type, including integers, strings, and even objects. Maps are widely used in Dart for tasks such as counting occurrences, grouping data, and representing complex relationships between entities. In this lab, students will learn how to create and manipulate maps in Dart to perform common programming tasks. They will explore various methods and operations provided by the Map class, including adding and removing key-value pairs, accessing values by keys, iterating over map entries, and more. Through a series of exercises and coding tasks, students will gain hands-on experience in working with maps and develop a deeper understanding of their functionality and versatility.

Maps are like dictionary like data types that exist in key-value form known as lock-key. There is no restriction on the type of data that goes in a map data type. Maps are flexible and can mutate their size based on their requirements

**Importance of Maps:**

Maps are essential data structures in Dart, as they provide a way to associate keys with values, allowing efficient retrieval and manipulation of data. Here are several reasons why maps are important in Dart: **Key-Value Pairing:** Maps allow you to store data in key-value pairs. This pairing enables you to retrieve values based on their associated keys quickly.

**Flexibility:** Maps can contain keys and values of different data types, providing flexibility in storing and organizing heterogeneous data.

**Efficient Data Retrieval:** Maps offer constant-time retrieval of values based on their keys. This efficiency is crucial when working with large datasets, as it allows for fast lookup operations.

**Data Organization:** Maps help in organizing data in a structured manner. You can use maps to represent relationships between entities, such as user profiles, product details, or configuration settings.

**Iterating Over Data:** Dart provides convenient methods for iterating over maps, allowing you to process each key-value pair efficiently. This feature is beneficial for performing operations on map data, such as filtering, mapping, or transforming values.

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

**JSON Parsing:** Maps are commonly used to represent JSON (JavaScript Object Notation) data in Dart. JSON data often consists of nested key-value pairs, making maps a natural choice for representing and manipulating JSON data structures.

**Communication with External Systems:** When interacting with external systems or APIs that use key value data formats, such as HTTP requests and responses, maps are used extensively for representing and processing data exchanged between the Dart application and external sources.

**State Management:** In Flutter, Dart's popular framework for building cross-platform mobile applications, maps are frequently used for managing application state. Flutter widgets often use maps to store and update component properties, such as widget configuration and state information.

Overall, maps play a crucial role in Dart programming, providing a powerful and versatile tool for storing, organizing, and manipulating data in a wide range of applications and scenarios.

**Creating Maps in Dart:**

import 'dart:io';

void main() {

var map\_name={

'key1':'sheerina',

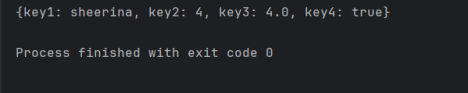
'key2':4,

'key3':4.0,

'key4':true};

print(map\_name);

}

**Access to individual value from a Map:**

import 'dart:io';

void main() {

var map\_name={

'key1':'sheerina',

'key2':4,

'key3':4.0,

'key4':true};

print(map\_name);

print(map\_name['key2']);

}

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487) 

If a key doesn’t exist then null value will be returned

import 'dart:io';

void main() {

var map\_name={

'key1':'sheerina',

'key2':4,

'key3':4.0,

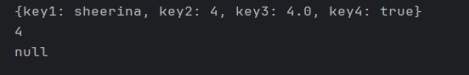
'key4':true};

print(map\_name);

print(map\_name['key2']);

print(map\_name['key5']);

}

The values in the maps can easily be updated as well and a new key:value pair can be added as well.

import 'dart:io';

void main() {

var map\_name={

'key1':'sheerina',

'key2':4,

'key3':4.0,

'key4':true};

print(map\_name);

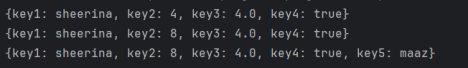
map\_name['key2']=8;

print(map\_name);

map\_name['key5']='maaz';

print(map\_name);

}

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487) A more realistic example of maps can be considered as employee record manipulation

import 'dart:io';

void main() {

var map\_name={

'NAME':'sheerina',

'YEAR\_OF\_EXPERIENCE':4,

'AVERAGE\_RATING':4.0,

'CAN\_LOCATE\_OFFICE':true};

print(map\_name);

}



Maps can even be declared and assigned after wards which assists in fetching real time data from the server.

import 'dart:io';

void main() {

var map\_name={

'NAME':'sheerina',

'YEAR\_OF\_EXPERIENCE':4,

'AVERAGE\_RATING':4.0,

'CAN\_LOCATE\_OFFICE':true};

print(map\_name);

var MapName=Map();

MapName['NAME']='Maaz';

MapName['YEAR\_OF\_EXPERIENCE']=4;

MapName['AVERAGE\_RATING']=3;

MapName['CAN\_LOCATE\_OFFICE']=true;

print(MapName);

}



DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

Some basic methods assist in making map manipulation efficient

import 'dart:io';

void main() {

var map\_name={

'NAME':'sheerina',

'YEAR\_OF\_EXPERIENCE':4,

'AVERAGE\_RATING':4.0,

'CAN\_LOCATE\_OFFICE':true};

print(map\_name);

var MapName=Map();

MapName['NAME']='Maaz';

MapName['YEAR\_OF\_EXPERIENCE']=4;

MapName['AVERAGE\_RATING']=3;

MapName['CAN\_LOCATE\_OFFICE']=true;

print(MapName);

print(MapName.isNotEmpty);

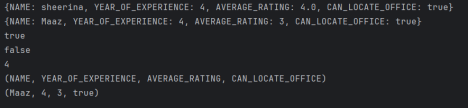
print(MapName.isEmpty);

print(MapName.length);

print(MapName.keys);

print(MapName.values);

}



|  |  |  |
| --- | --- | --- |
| 1 | .isNotEmpty | True if not empty |
| 2 | .IsEmpty | True if empty |
| 3 | .length | Returns Length of the Map |
| 4 | .keys | Returns all the keys |
| 5 | .values | Returns all the values |
| 6 | .containsKey | Search for a particular key |
| 7 | .containsValue | Search for a particular value |
| 8 | .remove | Remove a key:value pair |

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

import 'dart:io';

void main() {

var MapName=Map();

MapName['NAME']='Maaz';

MapName['YEAR\_OF\_EXPERIENCE']=4;

MapName['AVERAGE\_RATING']=3;

MapName['CAN\_LOCATE\_OFFICE']=true;

print(MapName);

print(MapName.isNotEmpty);

print(MapName.isEmpty);

print(MapName.length);

print(MapName.keys);

print(MapName.values);

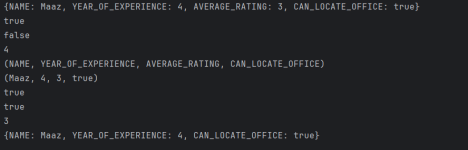
print(MapName.containsKey('CAN\_LOCATE\_OFFICE'));

print(MapName.containsValue(3));

print(MapName.remove('AVERAGE\_RATING'));

print(MapName);

}

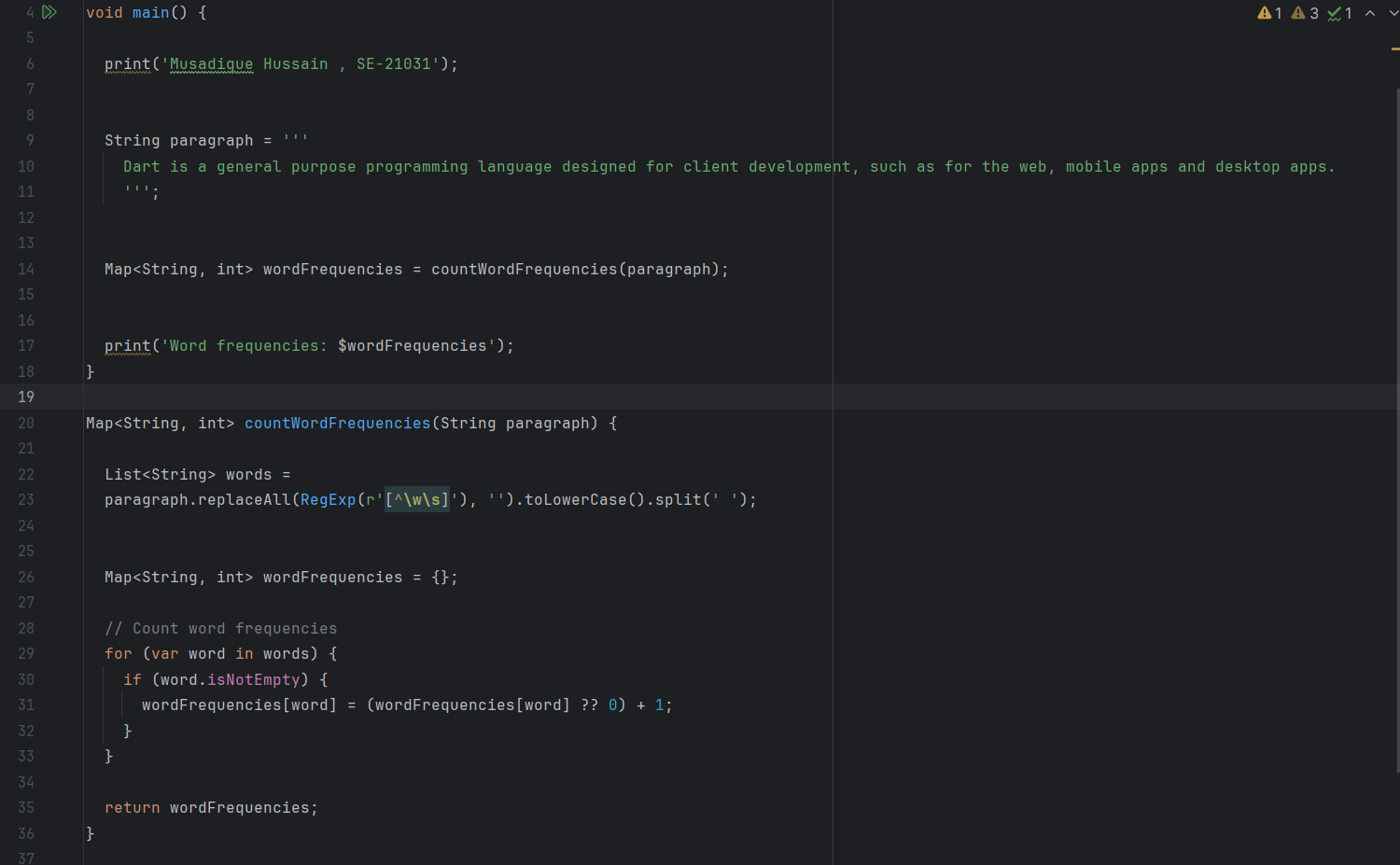


DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

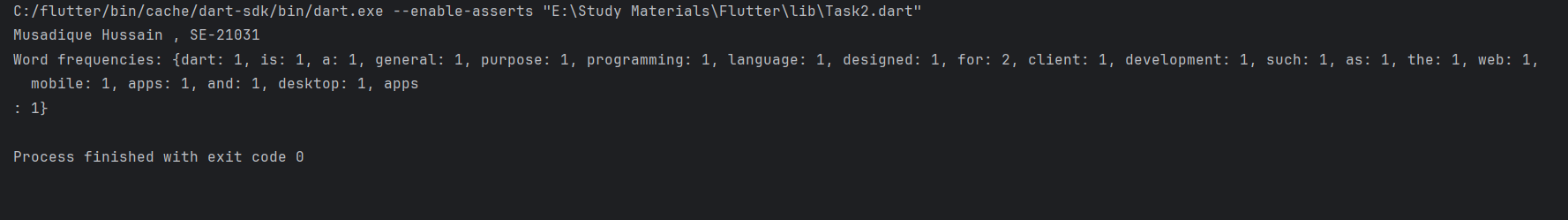
**Lab Exercise:**

**Question 1: Word Frequency Counter**

Given a paragraph of text, write a Dart function to count the frequency of each word and return the counts as a map.

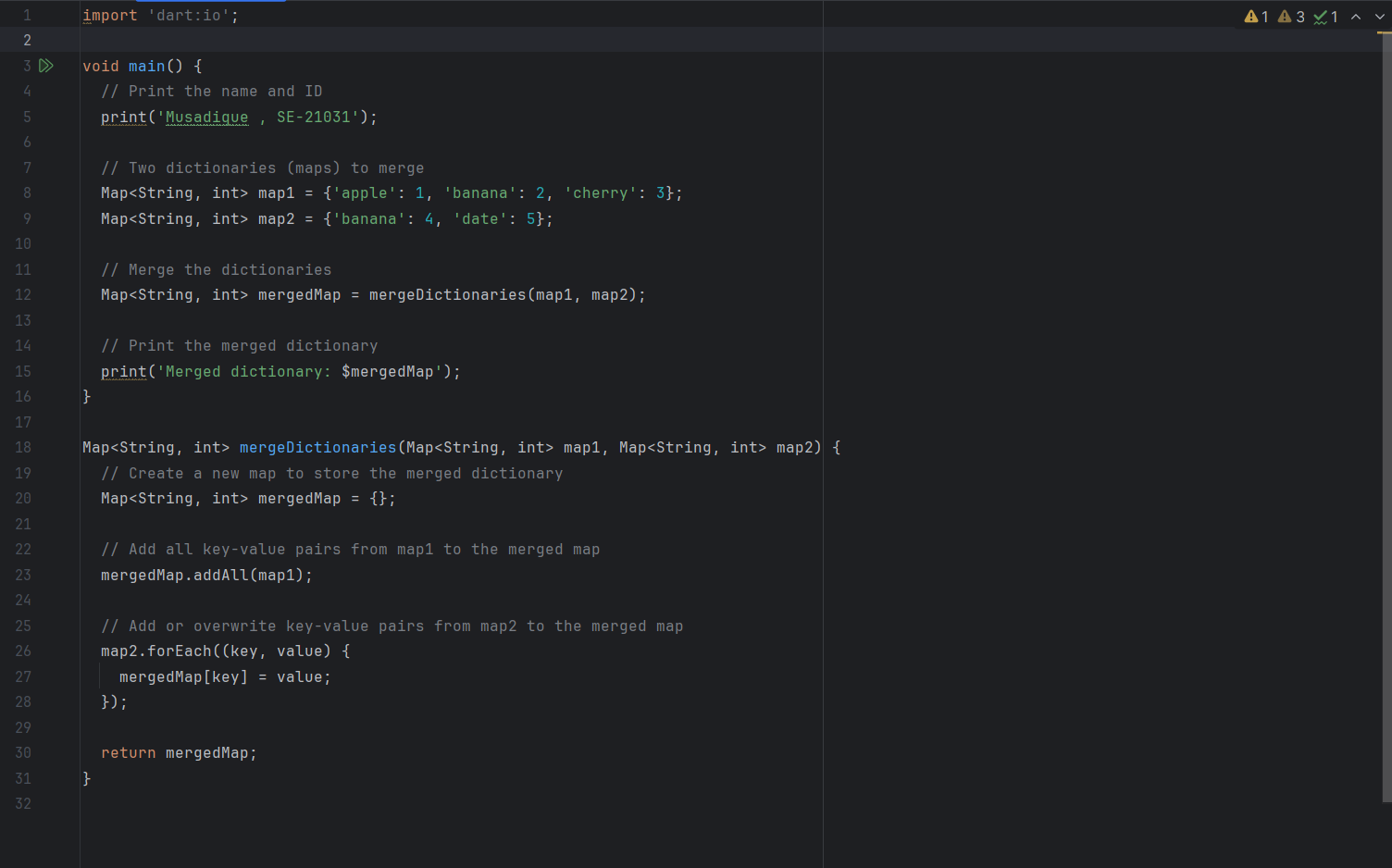


Output

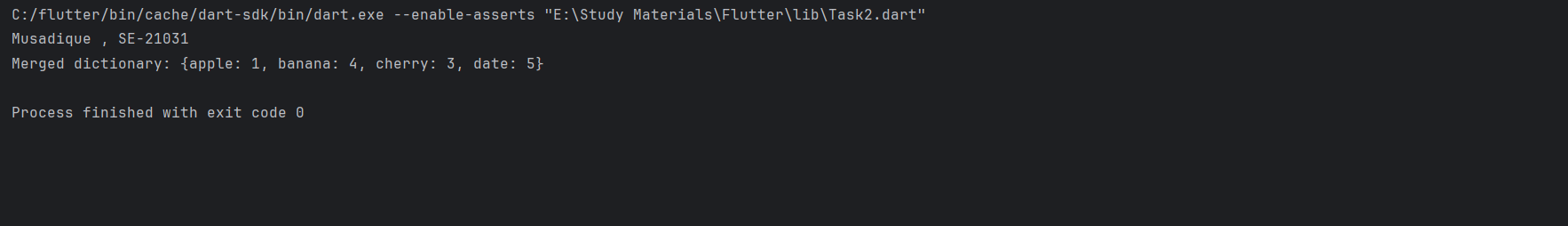


**Question 2: Merge Dictionaries**

Write a Dart function to merge two dictionaries (maps) into a single dictionary. If a key exists in both dictionaries, the value from the second dictionary should overwrite the value from the first dictionary.

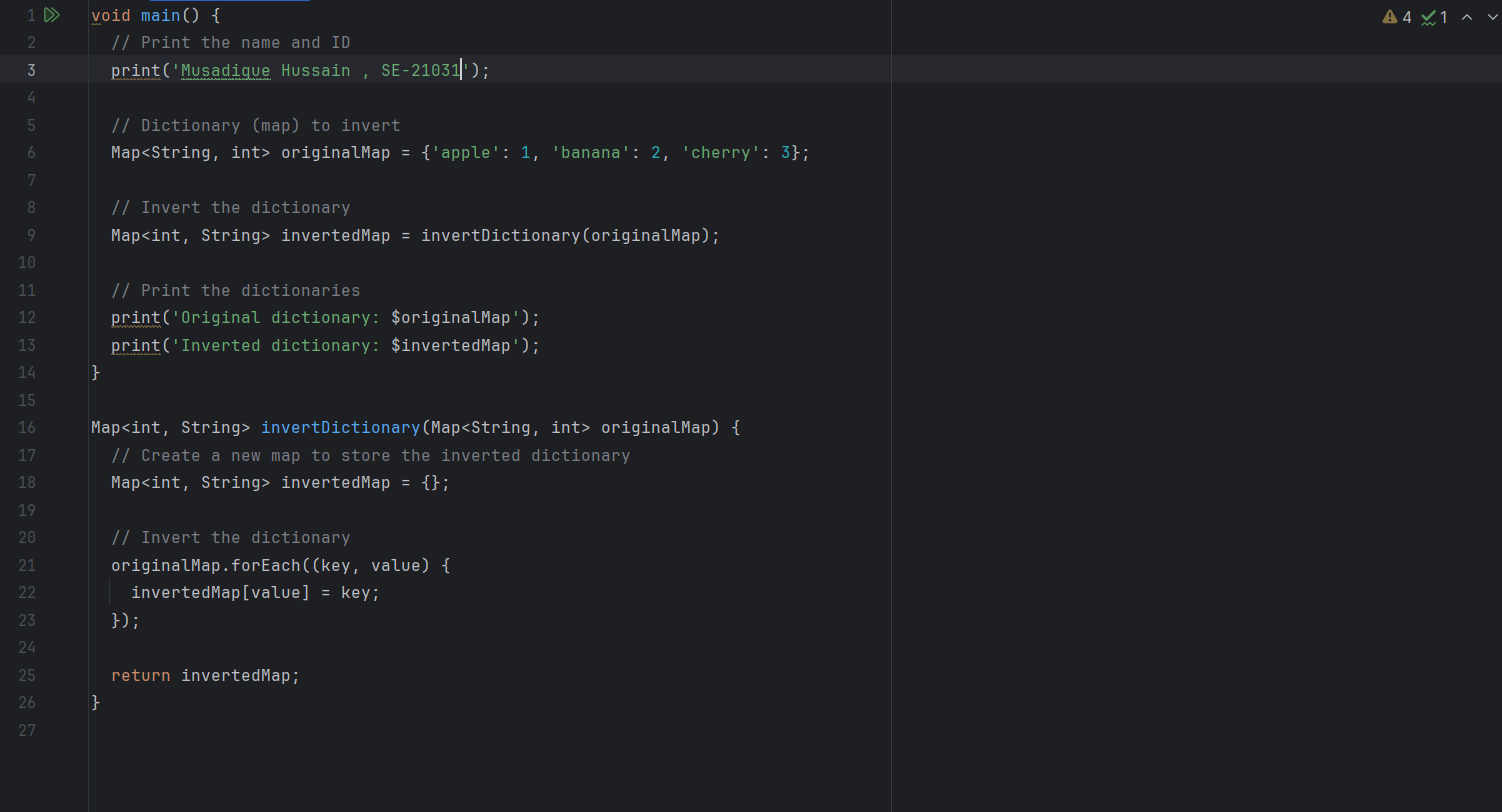


Output

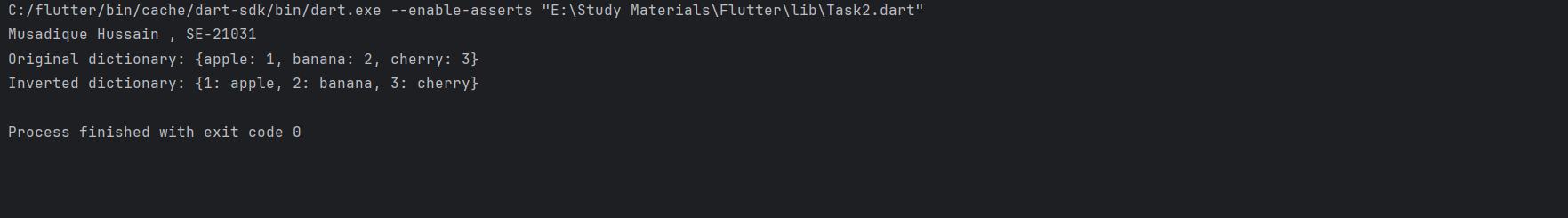


**Question 3: Invert Dictionary**

Write a Dart function to invert a dictionary (map), where the keys become the values and the values become the keys.

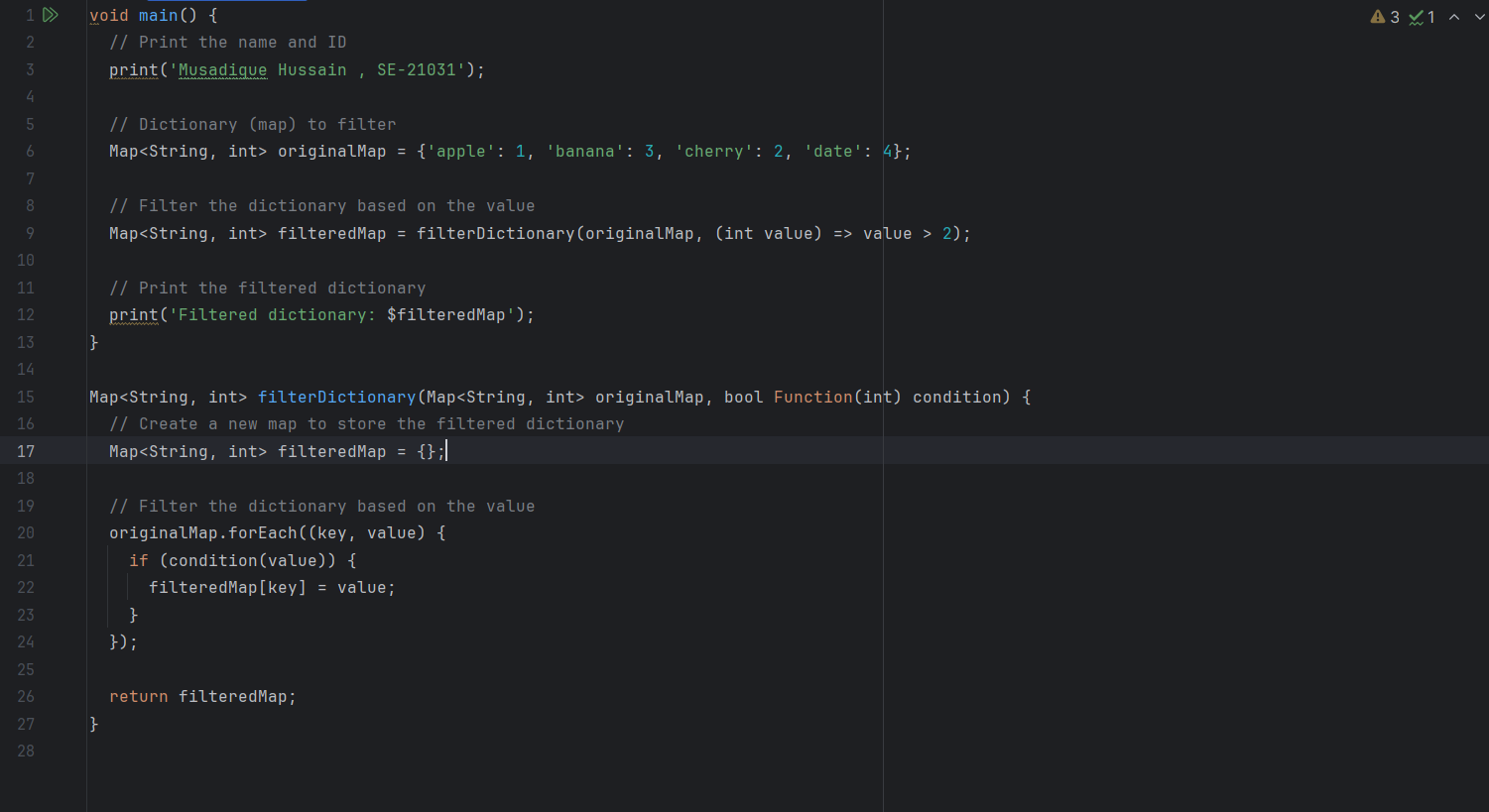


Output

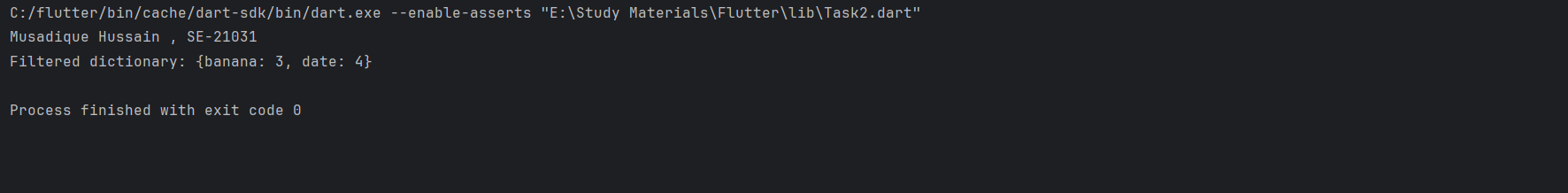


**Question 4: Filter Dictionary by Value**

Write a Dart function to filter a dictionary (map) based on the values. The function should return a new dictionary containing only key-value pairs where the value meets a specified condition.

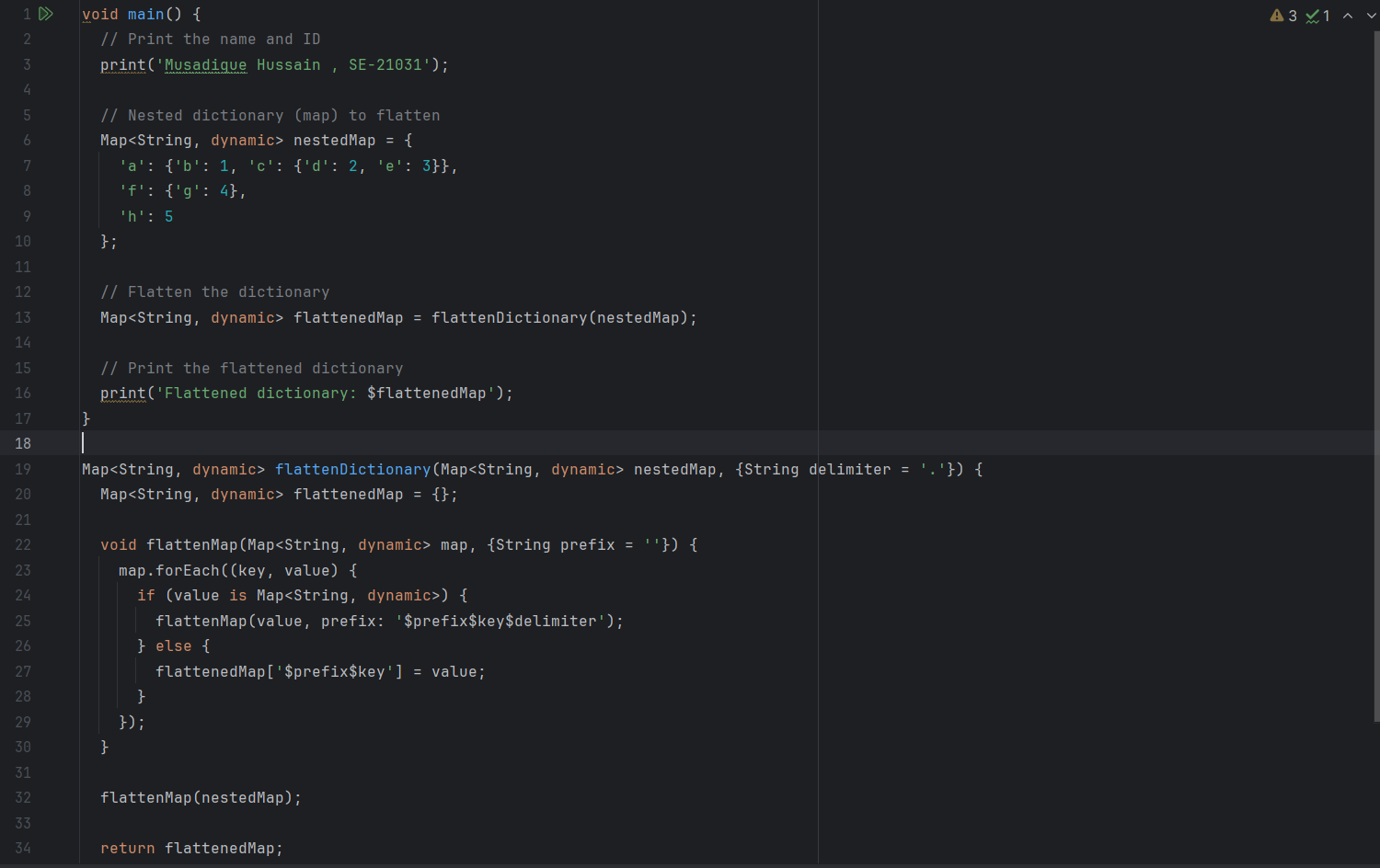


Output

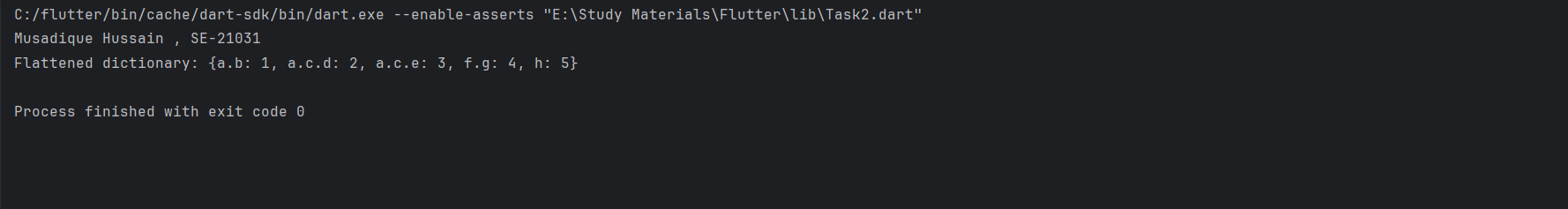


**Question 5: Flatten Dictionary**

Write a Dart function to flatten a nested dictionary (map). The function should convert nested keys into concatenated keys, separated by a delimiter.

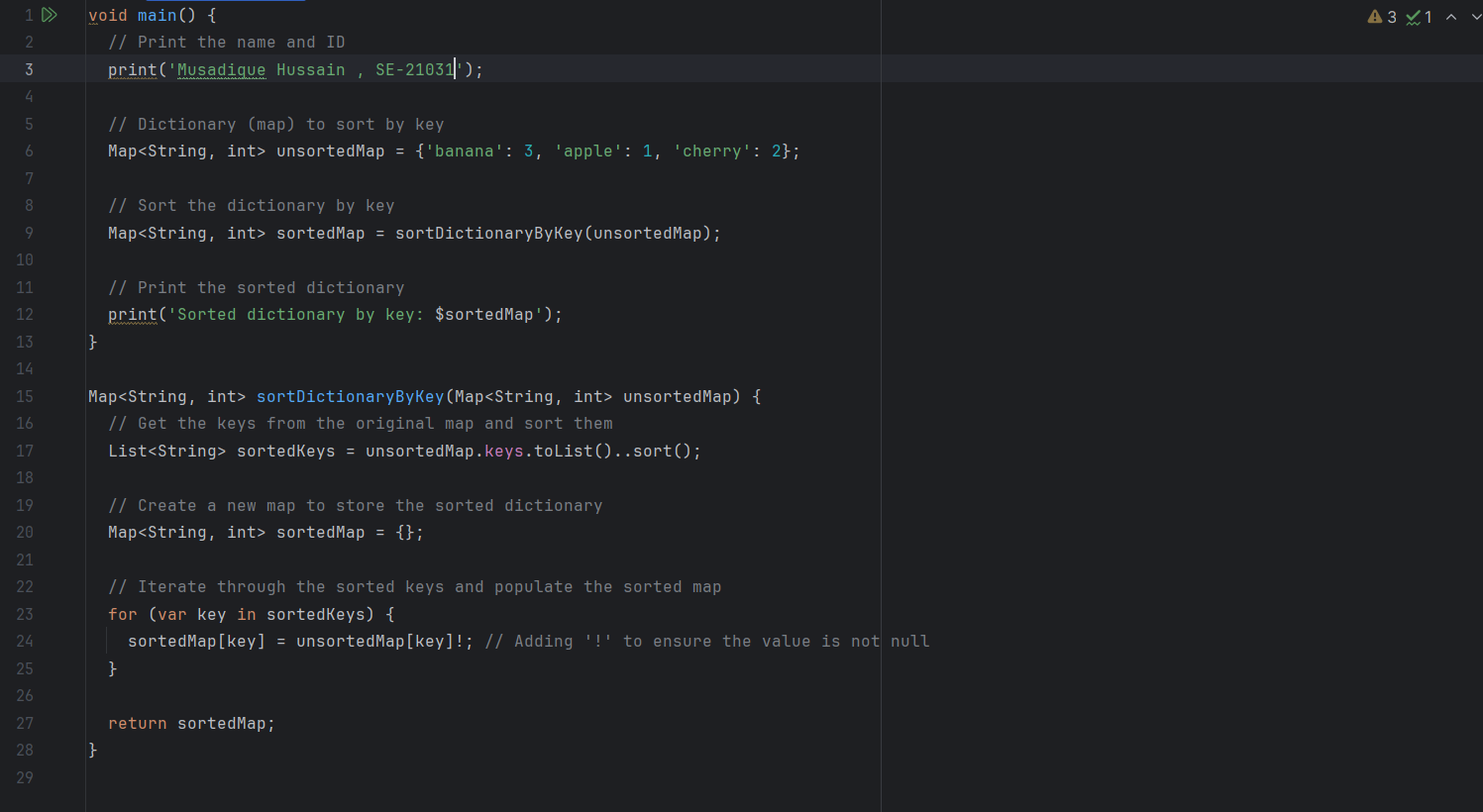


Output

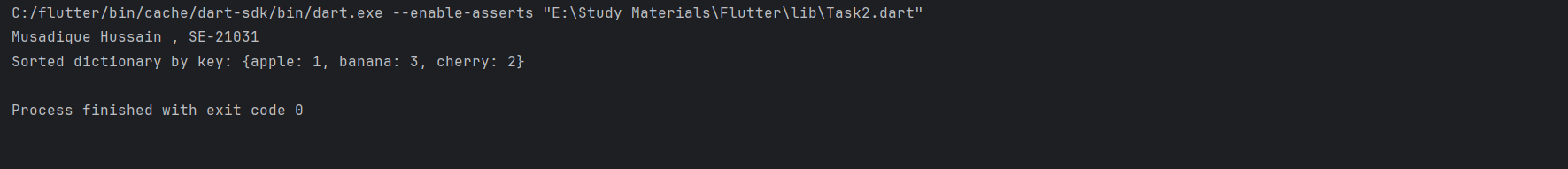


**Question 6: Sort Dictionary by Key**

Write a Dart function to sort a dictionary (map) by its keys in ascending order.

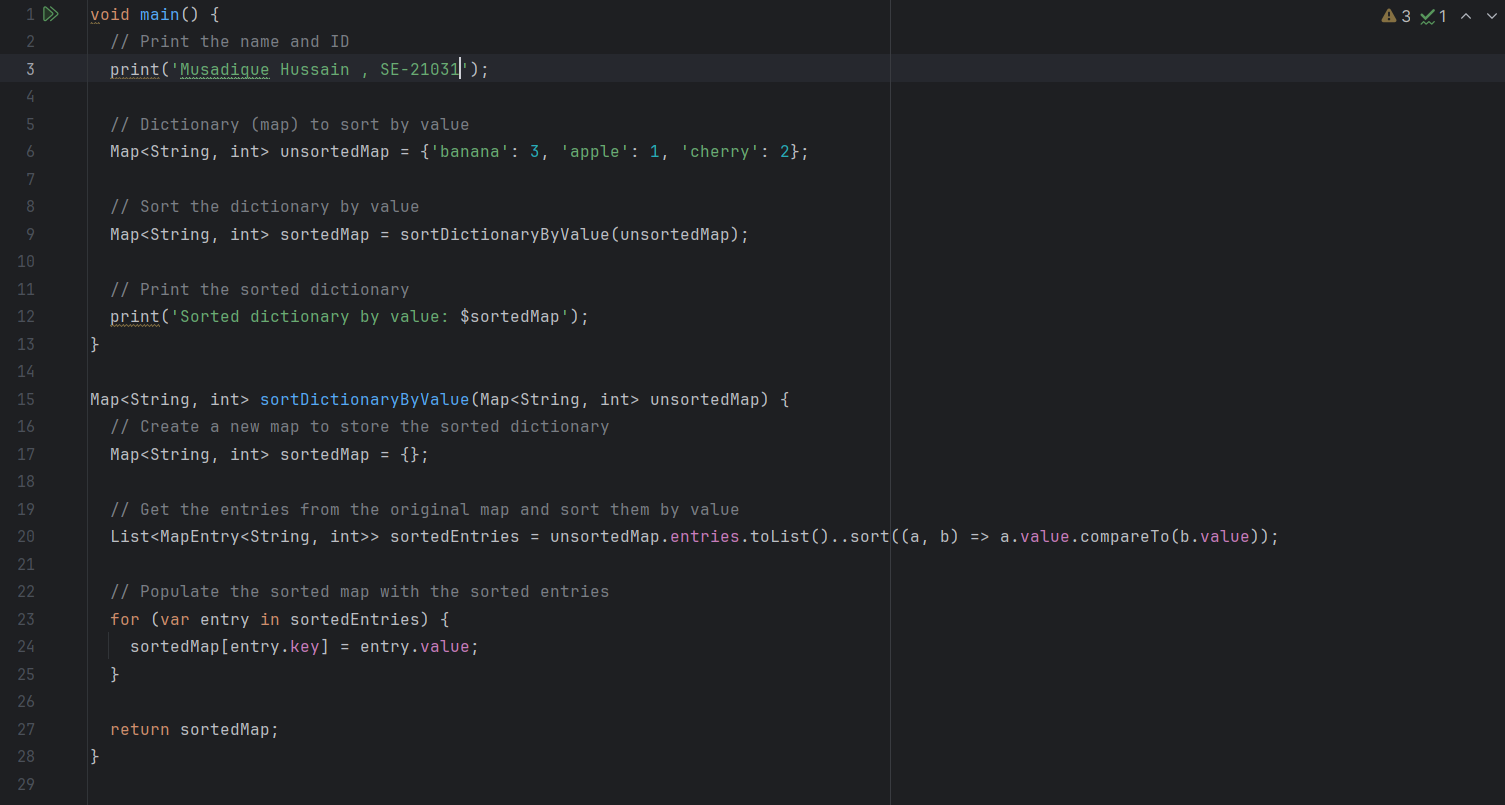


Output

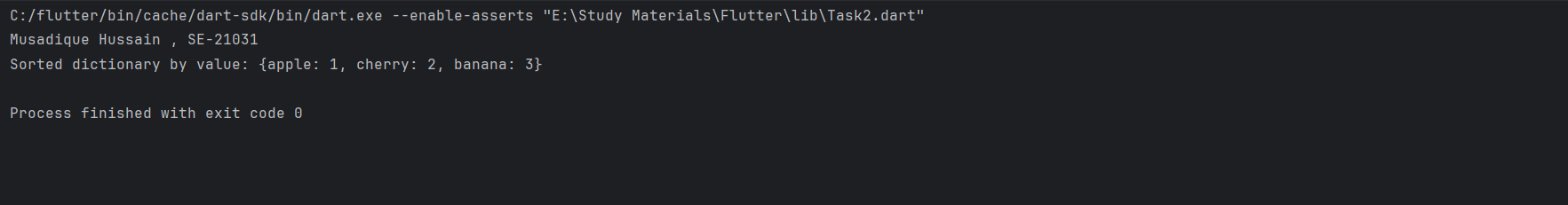


**Question 7: Sort Dictionary by Value**

Write a Dart function to sort a dictionary (map) by its values in ascending order.

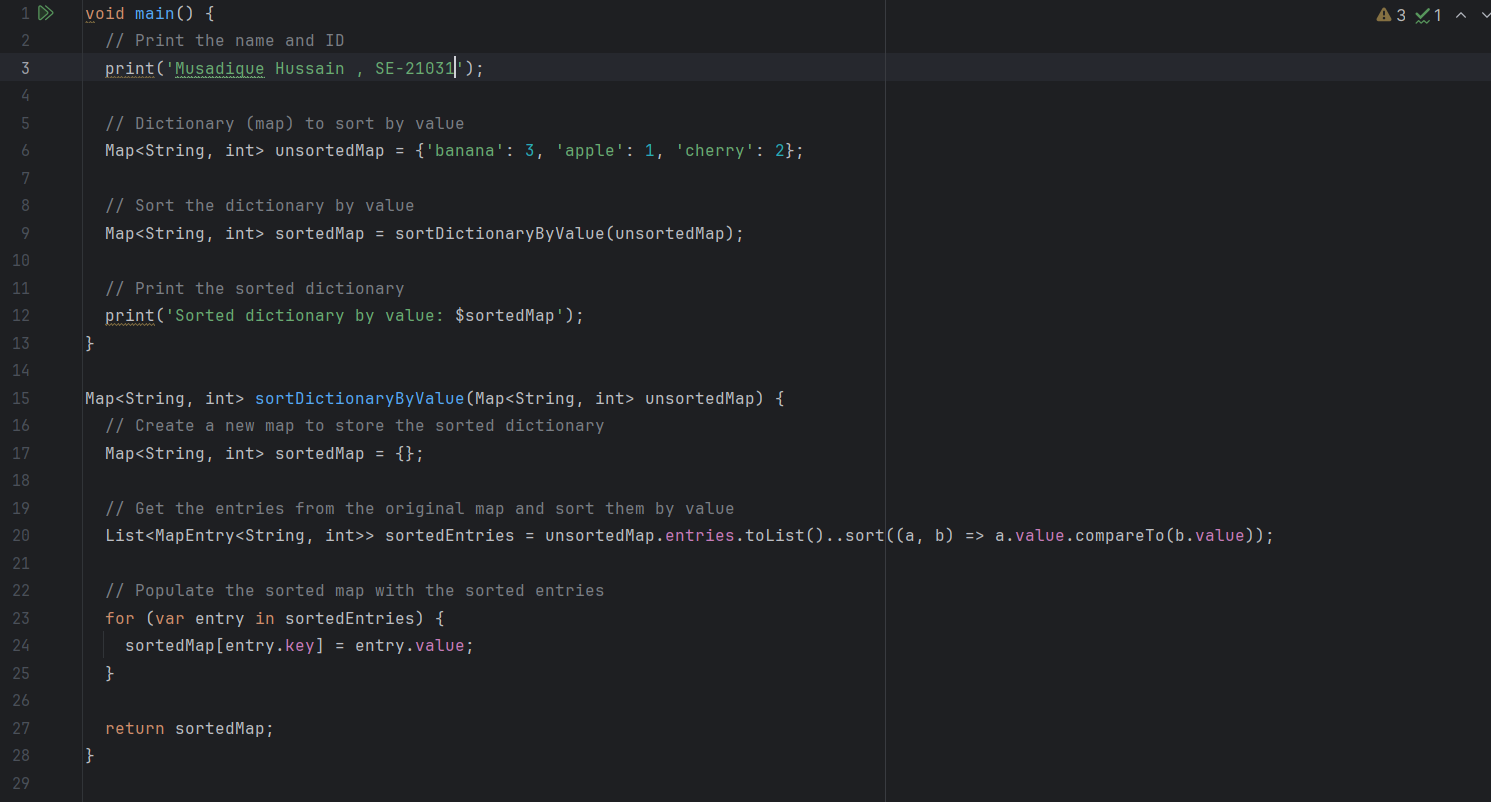


Output

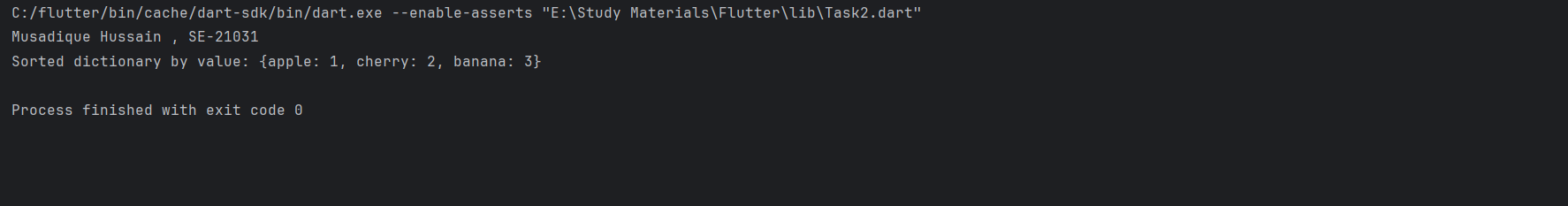


**Question 8: Group Anagrams**

Given a list of strings, write a Dart function to group anagrams together. Anagrams are strings that contain the same characters but in a different order.

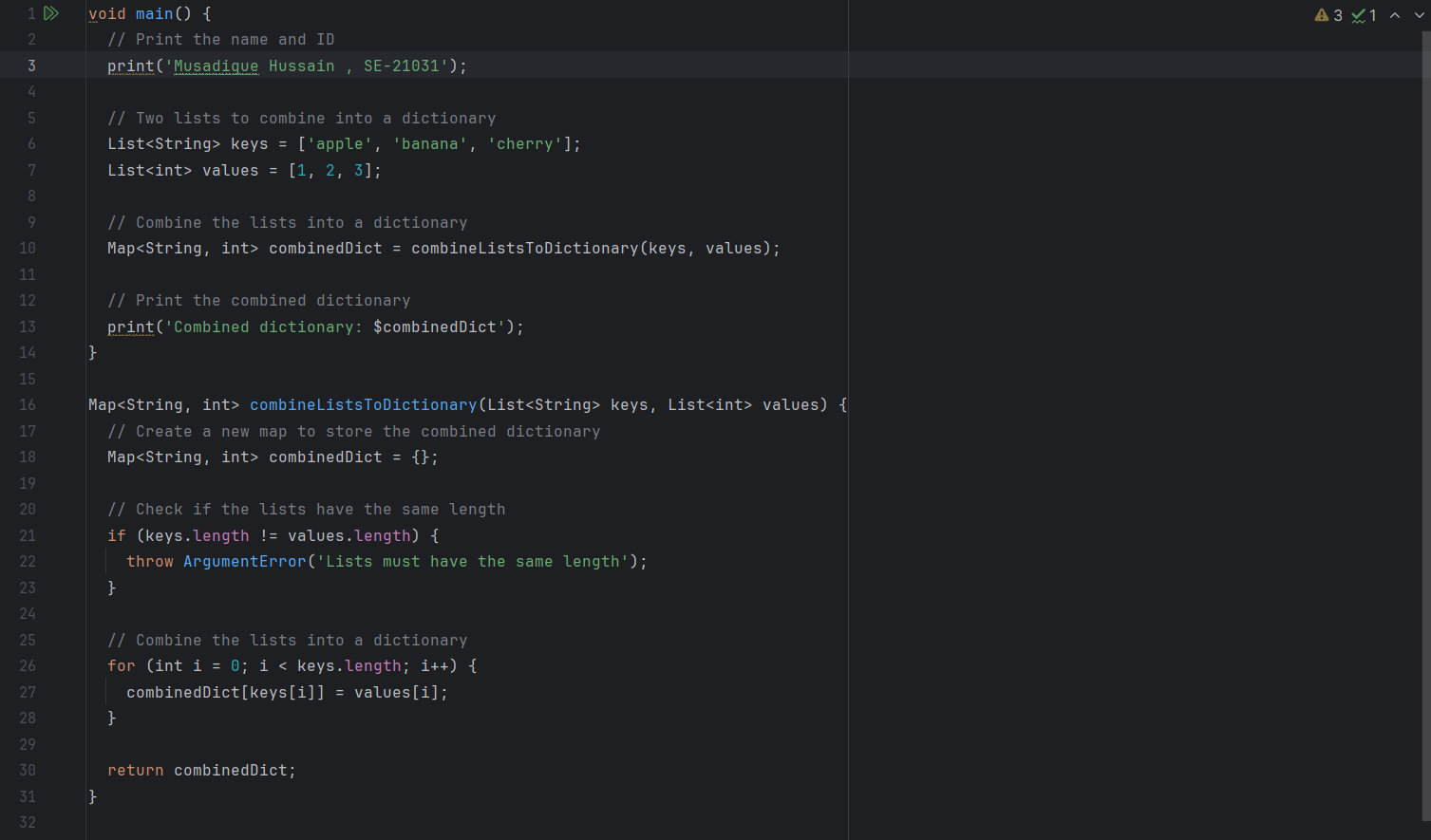


Output

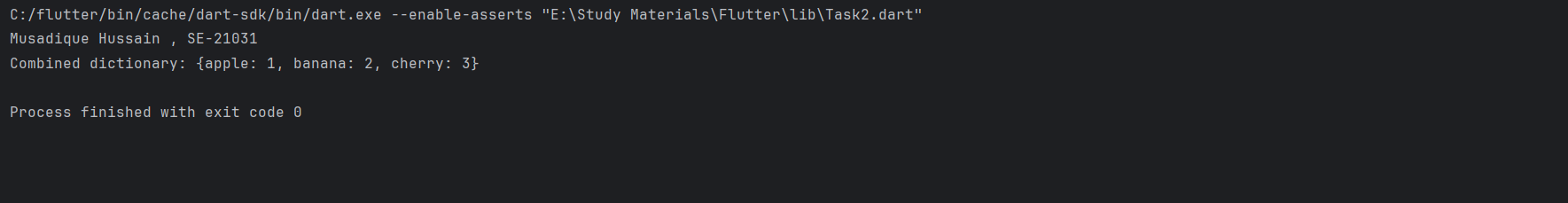


**Question 9: Combine Lists into Dictionary**

Write a Dart function to combine two lists into a dictionary (map), where the elements of one list serve as keys and the elements of the other list serve as values.

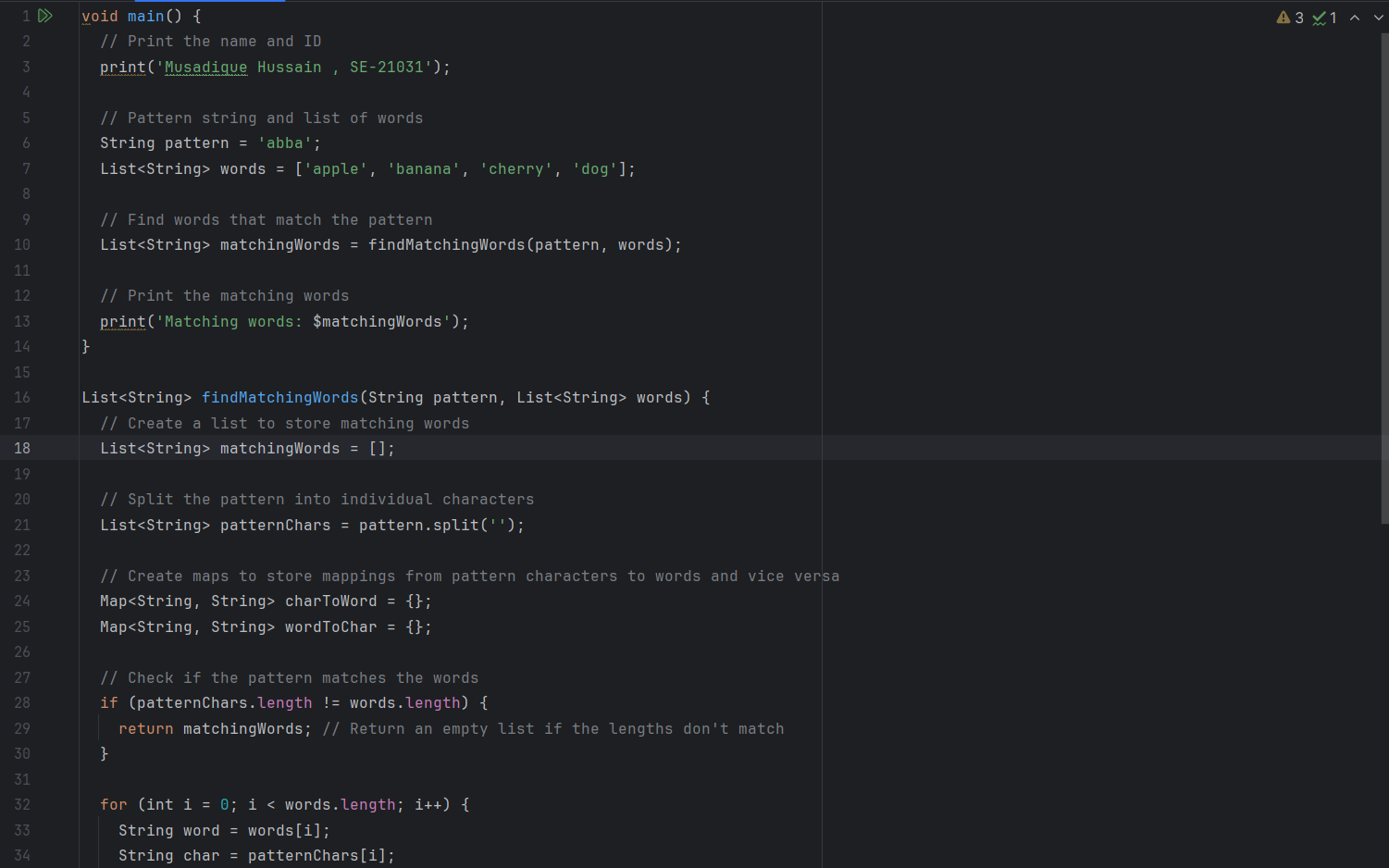


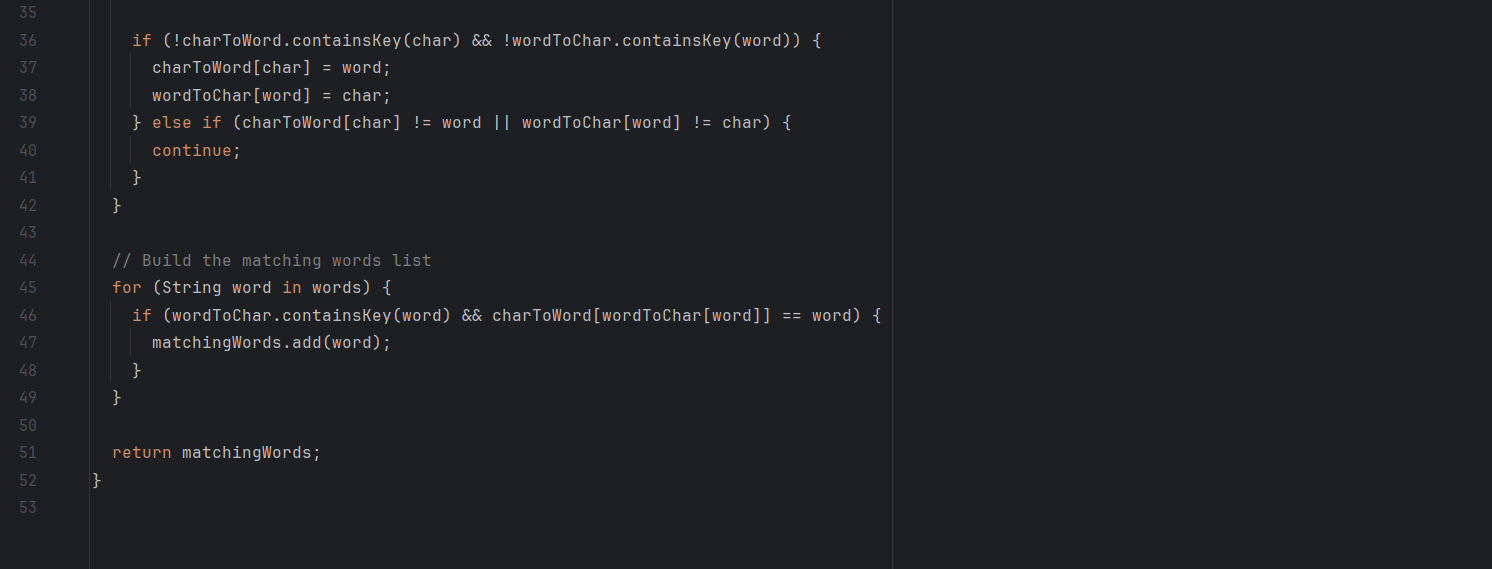
Output



**Question 10: Word Pattern Matching**

Given a pattern string and a list of words, write a Dart function to determine which words in the list match the pattern. The pattern consists of lowercase letters, where each letter corresponds to a unique word.





Output

