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

**LAB SESSION 3**

**LISTS IN DART PROGRAMMING**

**Objective**

The objective of this lab is to introduce you to the concept of lists in Dart programming language. You will learn how to declare, initialize, access elements, and perform basic operations on lists.

**Introduction**

Lists are basically ways of storing multiple data items just like an array .it can be a collection of similar data objects and different data objects as well. We already have studied about arrays in c language which is an ordered collection of objects. Array is the most commonly used collection in other programming languages whereas dart is used as a collection entity in dart.

**var list1=[10,20,30,40,50];**

Dart list is defined by storing all elements inside ([]) and separate the elements by ( ,).

List can be of fixed size (the size of the list can be fixed) and it can be grow able (here the items in the list are not fixed)

**Add Function**

In Dart, the add() function is a method provided by the List class, used to append an element to the end of an existing list. This method modifies the original list by adding the specified element as its last element. The syntax for using the add() function is straightforward:

List.add(element);

Where

List is the name of the list to which you want to add the element.

element is the value you want to add to the end of the list.

The add() function modifies the original list in place. It does not create a new list. The element added by add() becomes the last element of the list.

The add() function is commonly used when you need to dynamically add elements to a list, especially when the number of elements is not known in advance or when you want to append elements to the end of an existing list.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

Names.add(6);

print('$Names');

print('hello');

}

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



**Add All Function:**

In Dart, the addAll() function is used to append multiple elements from an iterable (such as another list) to the end of an existing list. This method modifies the original list by adding all the elements from the specified iterable to the end of the list. The syntax for using the addAll() function is as follows: **List.addAll(iterable);**

Where:

List is the name of the list to which you want to add elements.

iterable is an iterable object containing the elements you want to add to the list.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[];

Names.add(6);

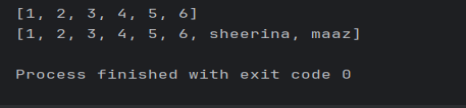
Names2.addAll(Names);

Names2.add('sheerina');

Names2.add('maaz');

print('$Names');

print('$Names2');



The addAll() function modifies the original list in place. It does not create a new list. The elements from the iterable are appended to the end of the list in the order they appear in the iterable.

The addAll() function is useful when you need to add multiple elements from another collection to an existing list, or when you want to concatenate two lists together. It provides a convenient way to extend the length of a list dynamically.

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

**Insert Function :**

In Dart, the insert() method is used to insert an element at a specified index within an existing list. This method modifies the original list by inserting the specified element at the specified index, shifting existing elements to the right. The syntax for using the insert() method is as follows:

List.insert(index, element);

Where:

List is the name of the list in which you want to insert the element.

index is the index at which you want to insert the element.

element is the value you want to insert into the list.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[];

Names.add(6);

Names2.addAll(Names);

Names2.add('sheerina');

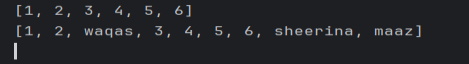
Names2.add('maaz');

Names2.insert(2,'waqas');

print('$Names');

print('$Names2');

}



The insert() method modifies the original list in place. It does not create a new list. The index specified must be within the bounds of the list. If the index is less than zero or greater than the length of the list, a Range Error will be thrown.

If the index is equal to the length of the list, the element will be added to the end of the list. The insert() method is useful when you need to insert an element at a specific position within a list, shifting the existing elements accordingly. It provides a flexible way to modify the contents of a list.

**Insert All Function:**

This function is applicable if it is intended to add one whole list at the end of another list starting from a certain index

. The syntax for using the insertAll() method is as follows:

List.insertAll(index,list2);

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

Where:

List is the name of the list in which you want to insert the element.

index is the index at which you want to insert the list.

List2 is the list you want to insert into the list.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[6,7,8,9,10];

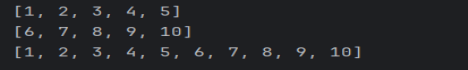
print('$Names');

print('$Names2');

Names.insertAll(5,Names2);

print("$Names");

}



**Update:**

If any value in the list needs to be added, the index shouls be known primarily.

List[index]=element

Where

List is the collection whose particular value is to be updated.

Index is the index of the element to be updated.

Element is the new value that is to be added.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[6,7,8,9,10];

print('$Names');

print('$Names2');

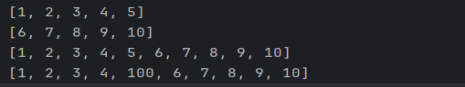
Names.insertAll(5,Names2);

print("$Names");

Names[4]=100;

print("$Names");

}

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

**Replace Range Function:**It is used when certain values of the lists are to be replaced. . The syntax for using replaceRange method is as follows:

List.replaceRange(start,end,list2);

Where

List is the list that is getting updated.

Start is the index from where replacement will start.

End is the index till which the replacement is to be done

List2 is the elements that are to be added in place of replacement.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[6,7,8,9,10];

print('$Names');

print('$Names2');

Names.insertAll(5,Names2);

print("$Names");

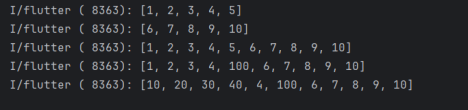
Names[4]=100;

print("$Names");

Names.replaceRange(0,3,[10,20,30,40]);

print("$Names");

}



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

**Removing An Element From The List:**

In Dart, you can remove elements from a list using various methods provided by the List class. Here's a summary of how you can use these methods:

**Remove Last Element:**

Use the removeLast() method to remove and return the last element of the list.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[6,7,8,9,10];

Names.insertAll(5,Names2);

Names[4]=100;

Names.replaceRange(0,3,[10,20,30,40]);

print("$Names");

Names.removeLast();

print("$Names");

}



**Remove Specific Element:**

Use the remove() method to remove the first occurrence of a specific element from the list.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[6,7,8,9,10];

Names.insertAll(5,Names2);

Names[4]=100;

Names.replaceRange(0,3,[10,20,30,40]);

print("$Names");

Names.removeLast();

print("$Names");

Names.remove(100);

print("$Names");

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

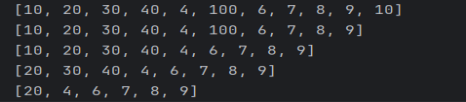
Names.removeAt(0);

print("$Names");

Names.removeRange(1,3);

print("$Names");

}

**Remove Element at Specific Index:**

Use the removeAt() method to remove the element at a specific index from the list.

var Names2=[6,7,8,9,10];

Names.insertAll(5,Names2);

Names[4]=100;

Names.replaceRange(0,3,[10,20,30,40]);

print("$Names");

Names.removeLast();

print("$Names");

Names.remove(100);

print("$Names");

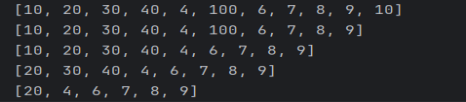
Names.removeAt(0);

print("$Names");

Names.removeRange(1,3);

print("$Names");

}

**Remove Elements within a Range:**

Use the removeRange function to remove elements within a particular range

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

var Names2=[6,7,8,9,10];

Names.insertAll(5,Names2);

Names[4]=100;

Names.replaceRange(0,3,[10,20,30,40]);

print("$Names");

Names.removeLast();

print("$Names");

Names.remove(100);

print("$Names");

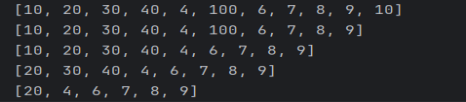
Names.removeAt(0);

print("$Names");

Names.removeRange(1,3);

print("$Names");

}



In Dart, you can use various methods and properties to work with lists effectively. Here's a summary of commonly used methods and properties for list manipulation:

isEmpty:Use the isEmpty property to check if the list is empty. It returns true if the list contains no elements, false otherwise.

isNotEmpty:Use the isNotEmpty property to check if the list is not empty. It returns true if the list contains one or more elements, false otherwise.

first:Use the first property to get the first element of the list.

last:Use the last property to get the last element of the list.

length:Use the length property to get the number of elements in the list.

reversed:Use the reversed property to get an iterable of the elements in the list in reverse order. elementAt:Use the elementAt() method to get the element at a specified index in the list.

import 'dart:io';

void main() {

var Names=[1,2,3,4,5];

var Names2=[6,7,8,9,10];

Names.insertAll(5,Names2);

Names[4]=100;

Names.replaceRange(0,3,[10,20,30,40]);

print("$Names");

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

print("EMPTY:${Names.isEmpty}");

print("NOT EMPTY:${Names.isNotEmpty}");

print("FIRST:${Names.first}");

print("LAST:${Names.last}");

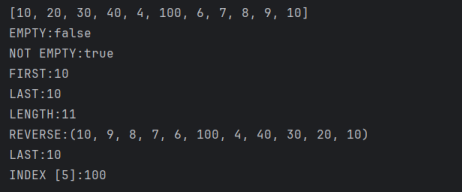
print("LENGTH:${Names.length}");

print("REVERSE:${Names.reversed}");

print("LAST:${Names.last}");

print("INDEX [5]:${Names.elementAt(5)}");

}



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

**Lab Exercise**

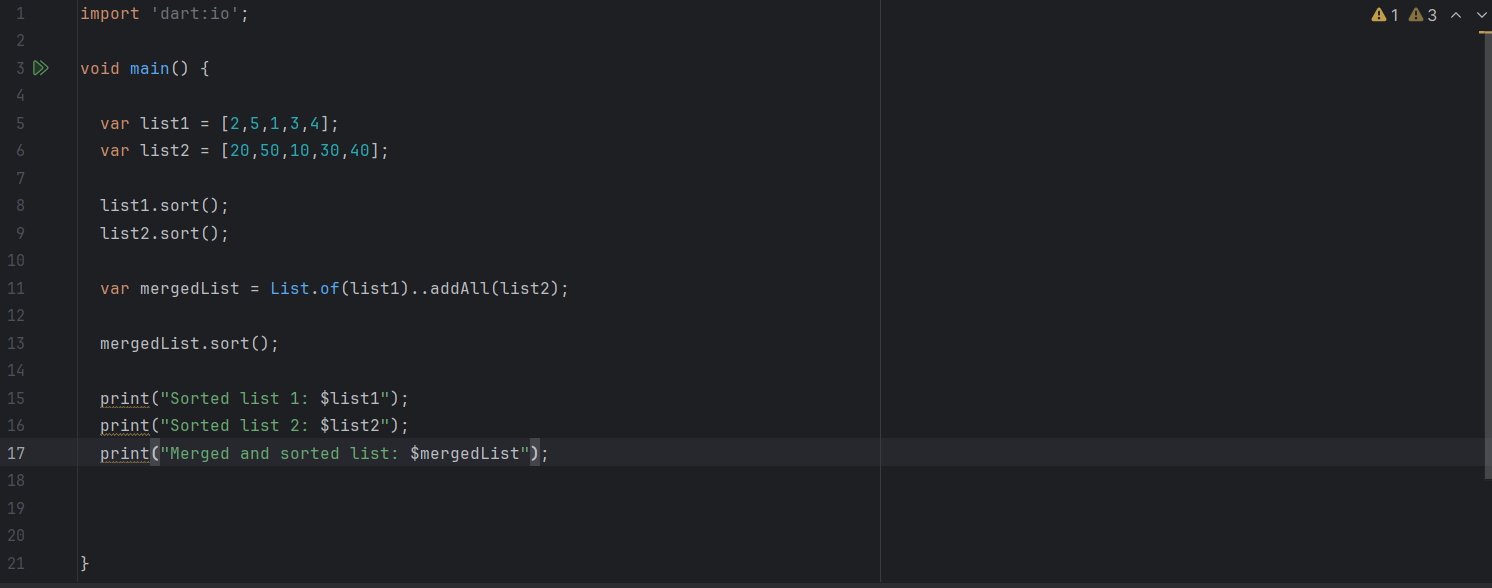
Attach the screen shots of the code and its output. Every code should have your name and roll number printed first in the program:

**Task 1: Sorting and Merging**

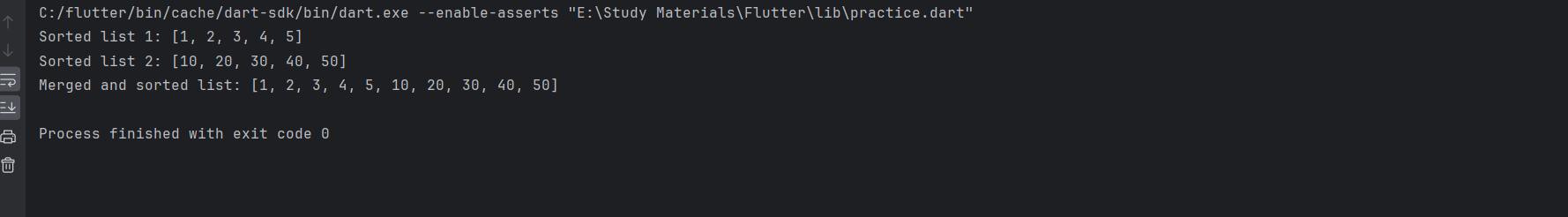
1. Create two lists of integers.

2. Sort both lists in ascending order.

3. Merge the two sorted lists into a single sorted list.



Output

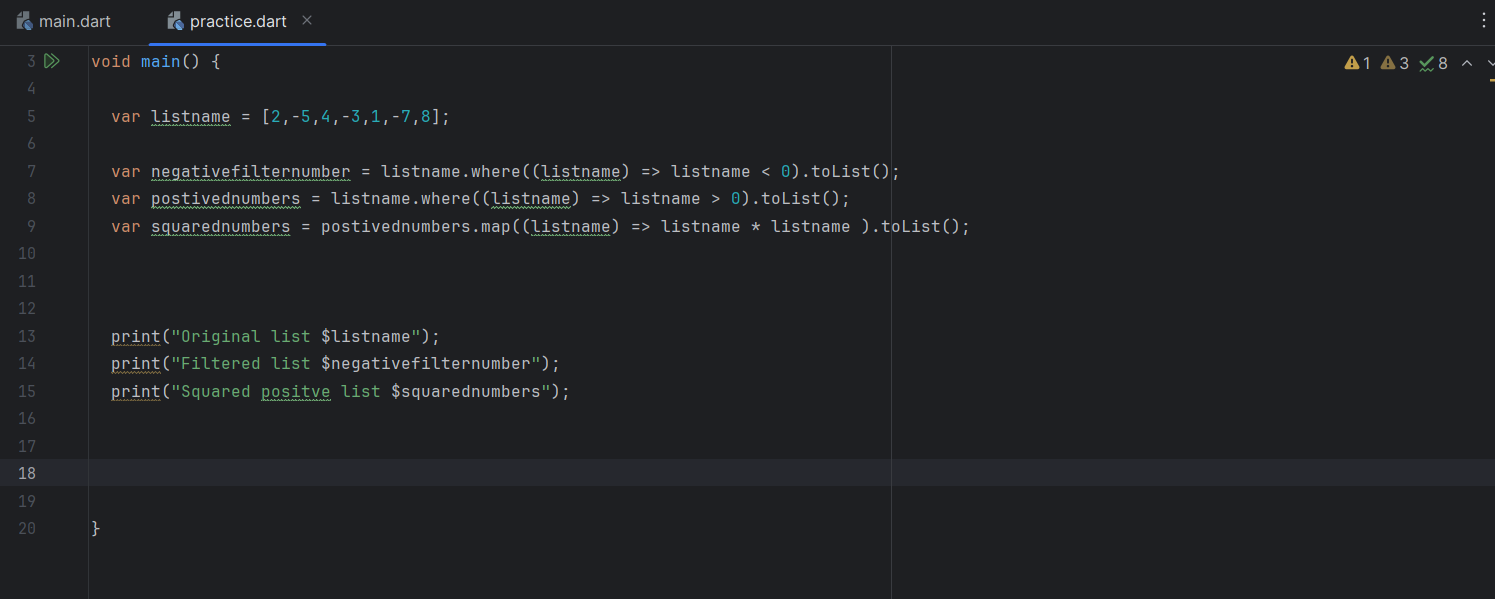


**Task 2: Filtering and Mapping**

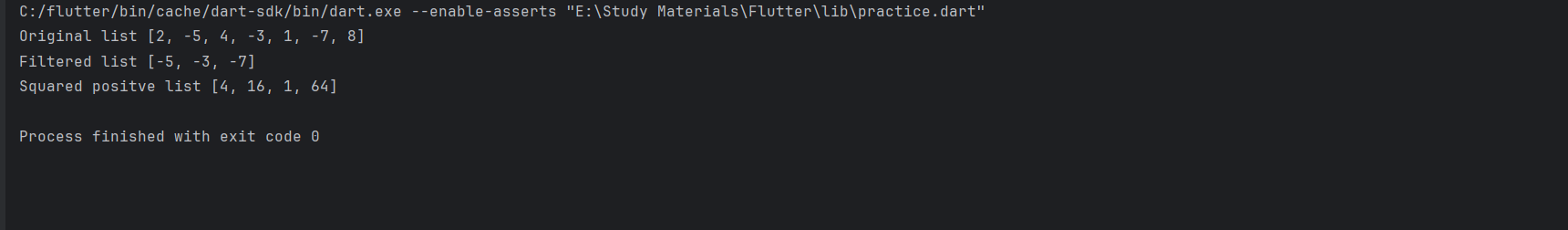
1. Create a list of integers containing both positive and negative numbers.

2. Filter out all negative numbers from the list.

3. Map the remaining positive numbers to their squares.



Output

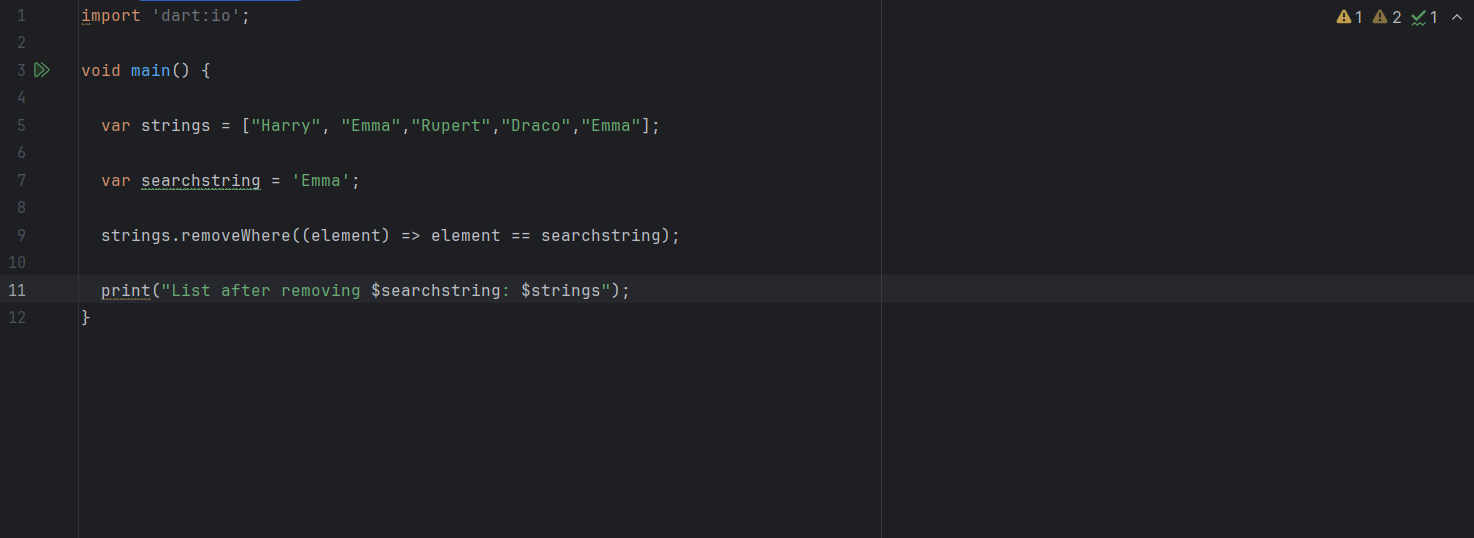


**Task 3: Searching and Removing**

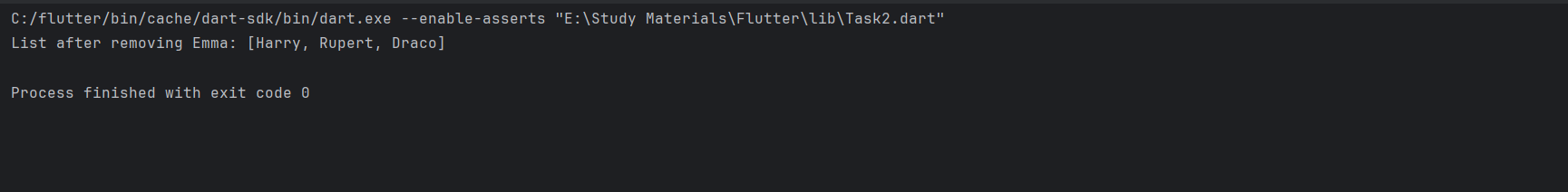
1. Create a list of strings containing duplicate elements.

2. Search for a specific string in the list.

3. Remove all occurrences of the searched string from the list.



Output

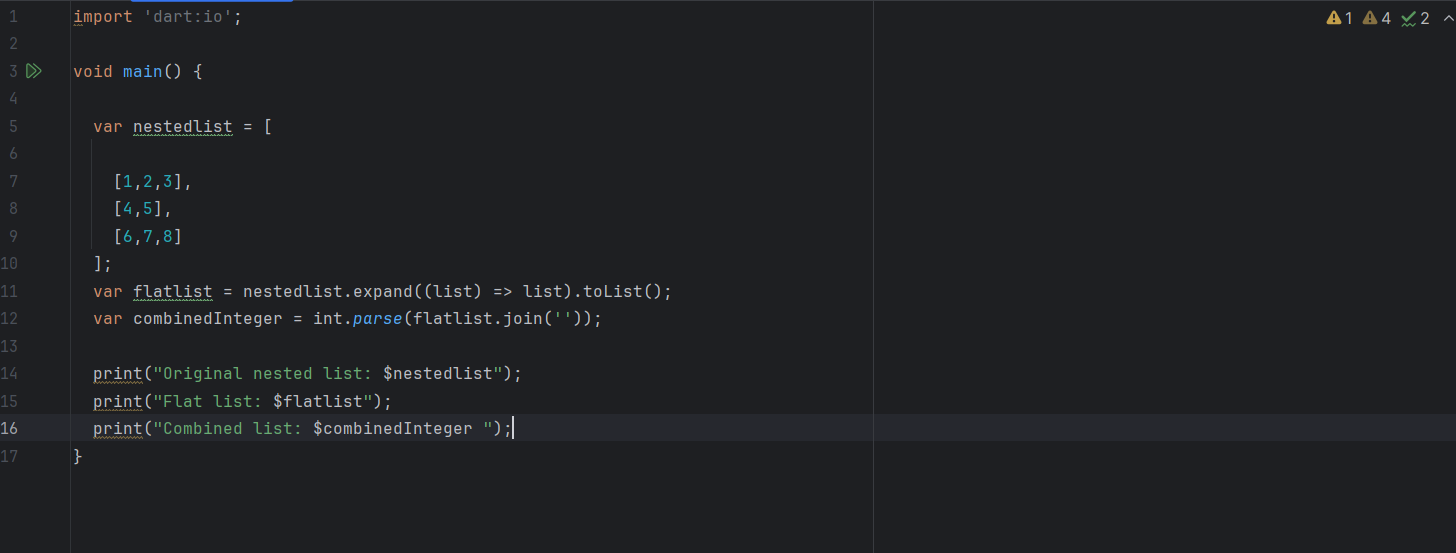


**Task 4: Extracting and Combining**

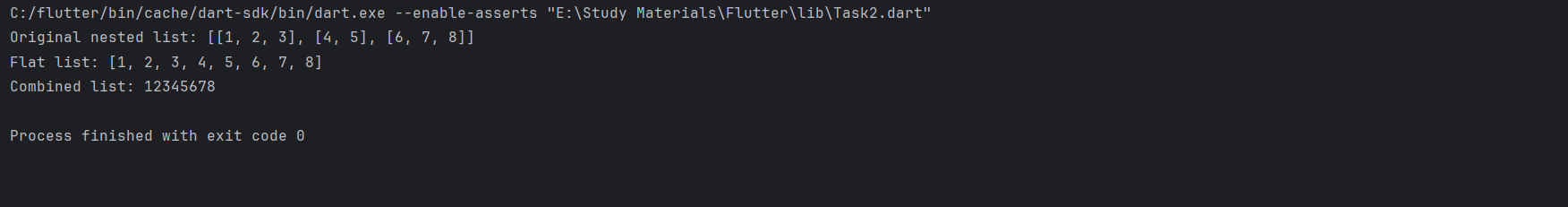
1. Create a list of lists, each containing integers.

2. Extract all integers from the nested lists into a single flat list.

3. Combine all integers from the flat list into a single integer.



Output



**Task 5: Reversing and Summing**

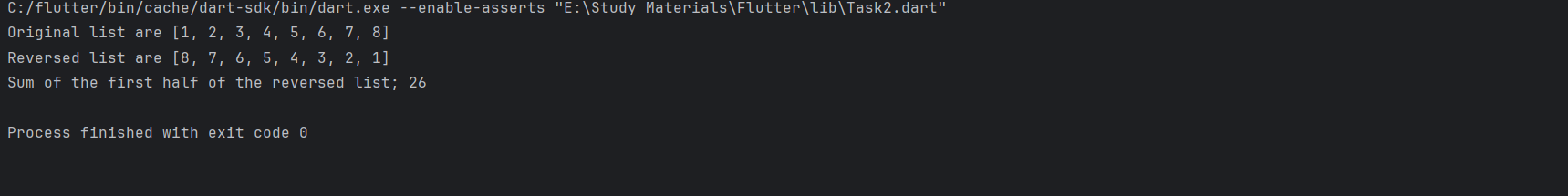
1. Create a list of integers.

2. Reverse the order of elements in the list.

3. Calculate the sum of the first half of the list.



Output



**Task 6: Sorting and Grouping**

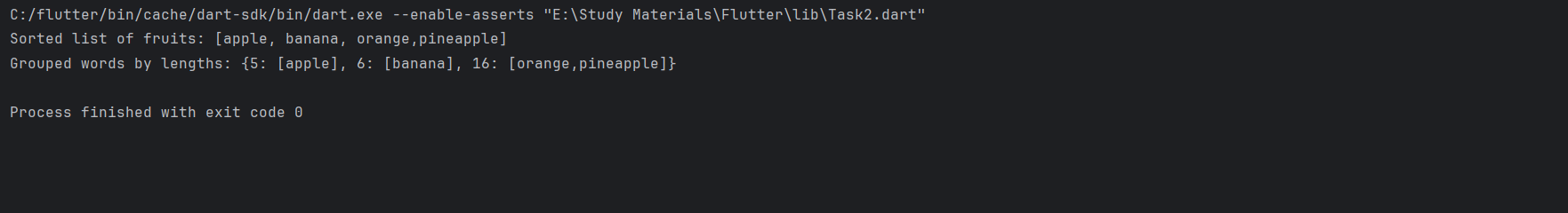
1. Create a list of strings containing words of varying lengths.

2. Sort the list based on the lengths of the words (shortest to longest).

3. Group the words in the sorted list by their lengths.



Output

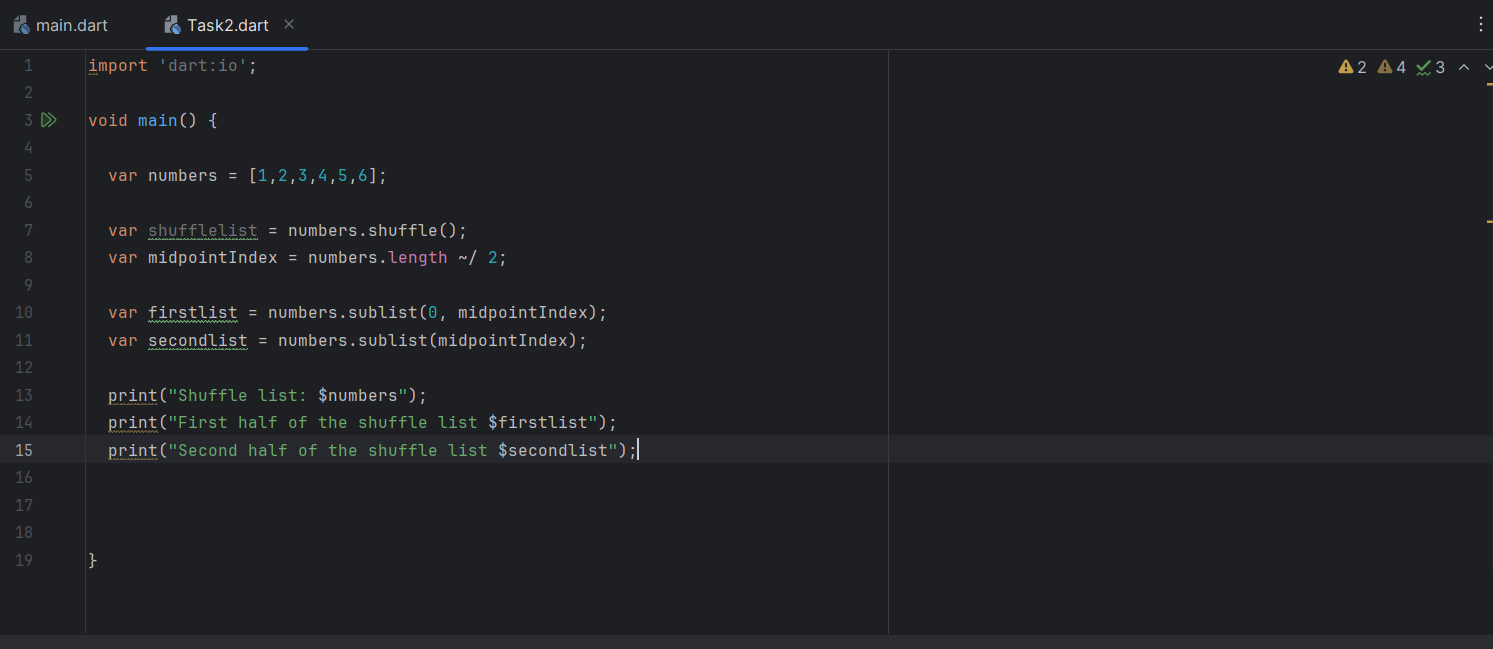


**Task 7: Shuffling and Slicing**

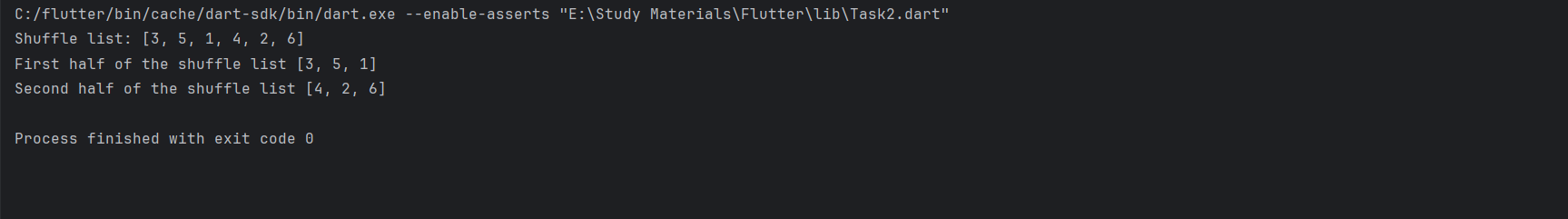
1. Create a list of integers.

2. Shuffle the elements of the list randomly.

3. Slice the shuffled list into two separate lists of equal size.



Output

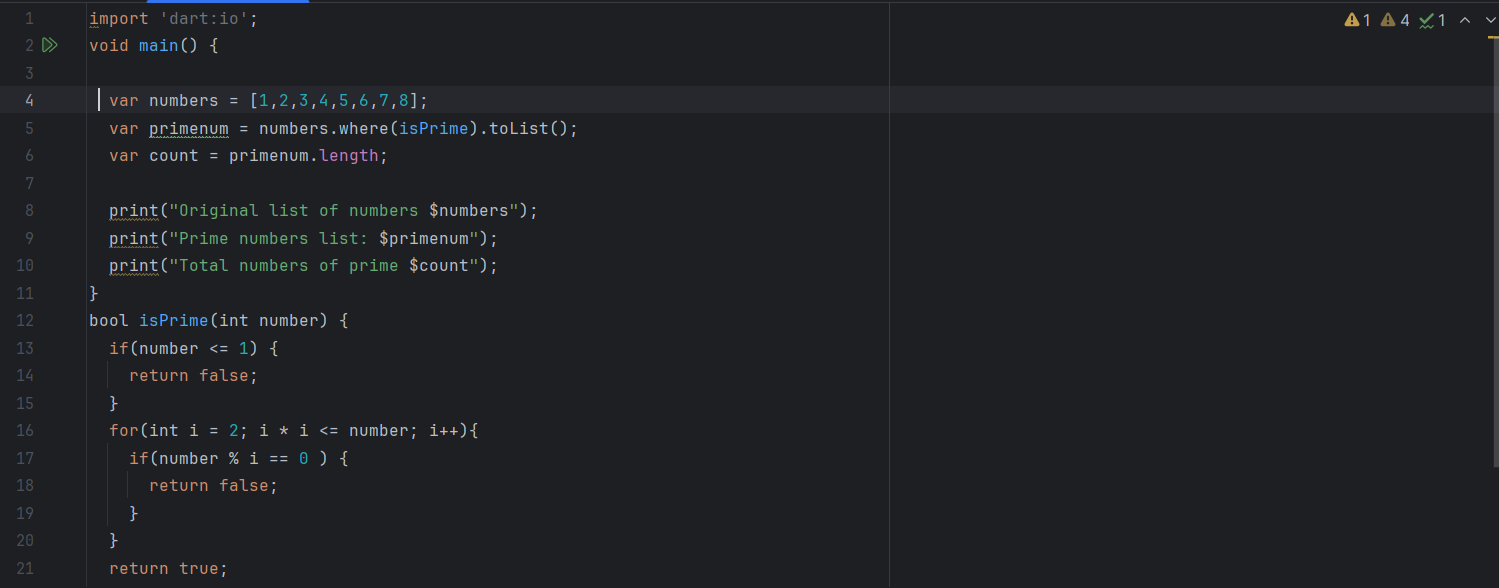


**Task 8: Filtering and Counting**

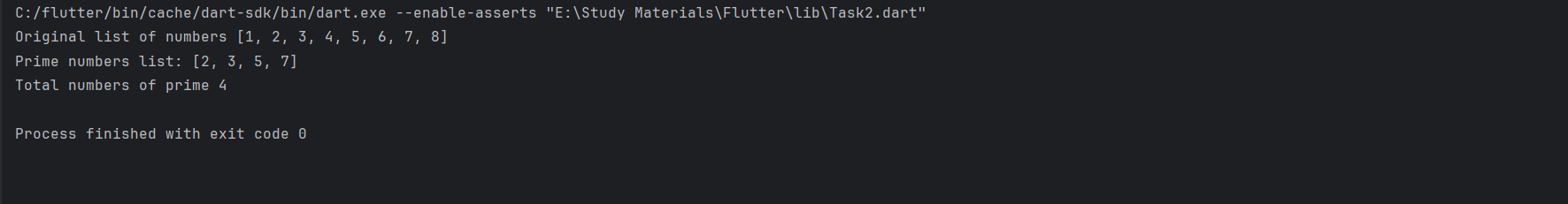
1. Create a list of integers.

2. Filter out all prime numbers from the list.

3. Count the total number of prime numbers remaining in the list.



Output



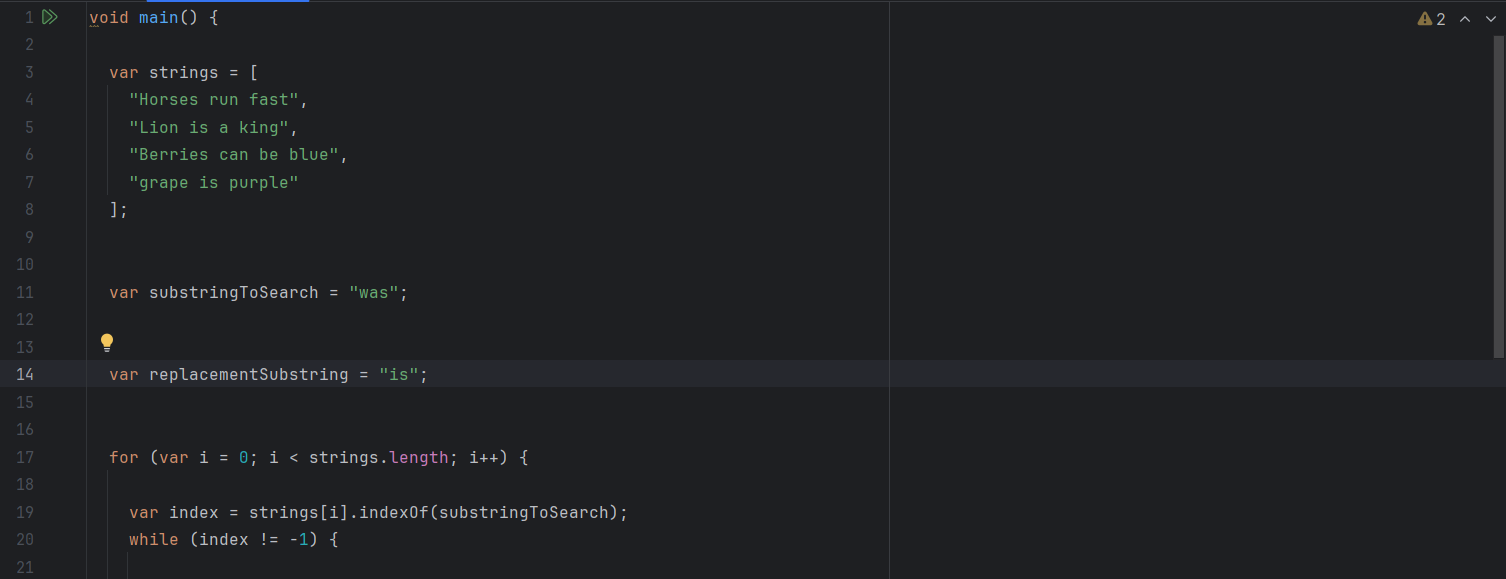
**Task 9: Searching and Substituting**

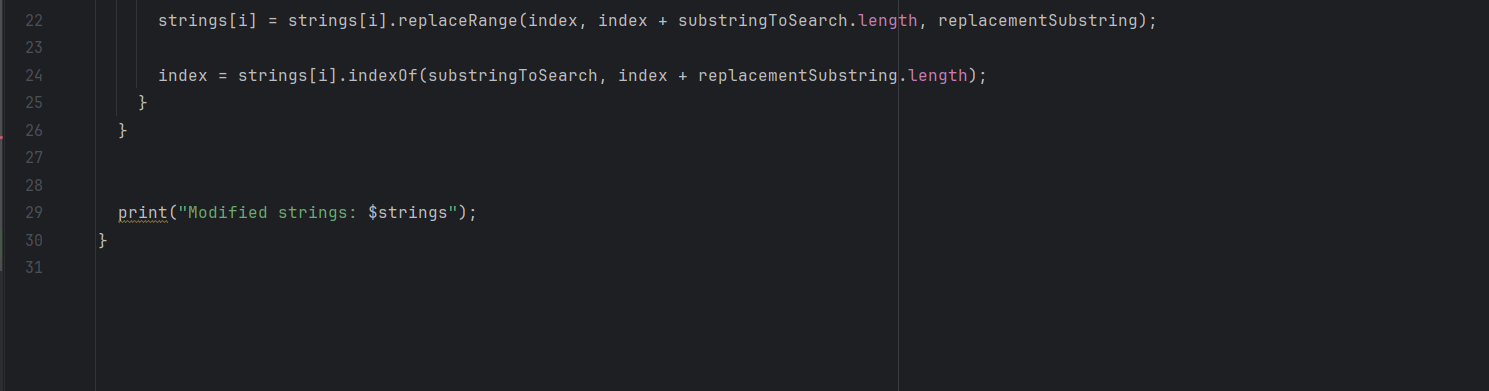
1. Create a list of strings.

2. Search for all occurrences of a specific substring in the strings.

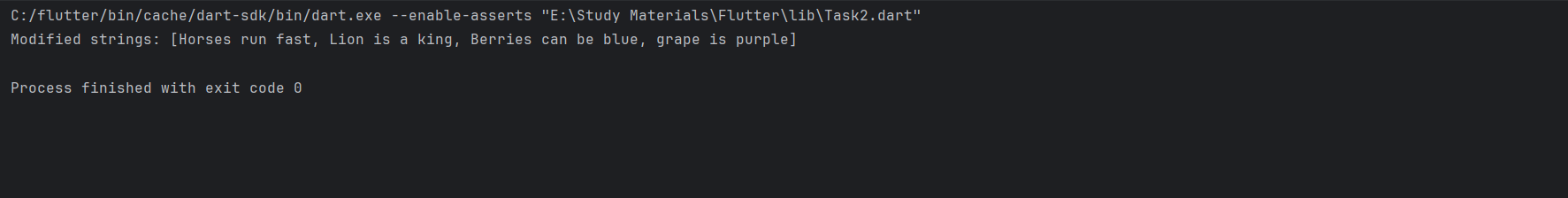
DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

3. Substitute the found substrings with a different substring.





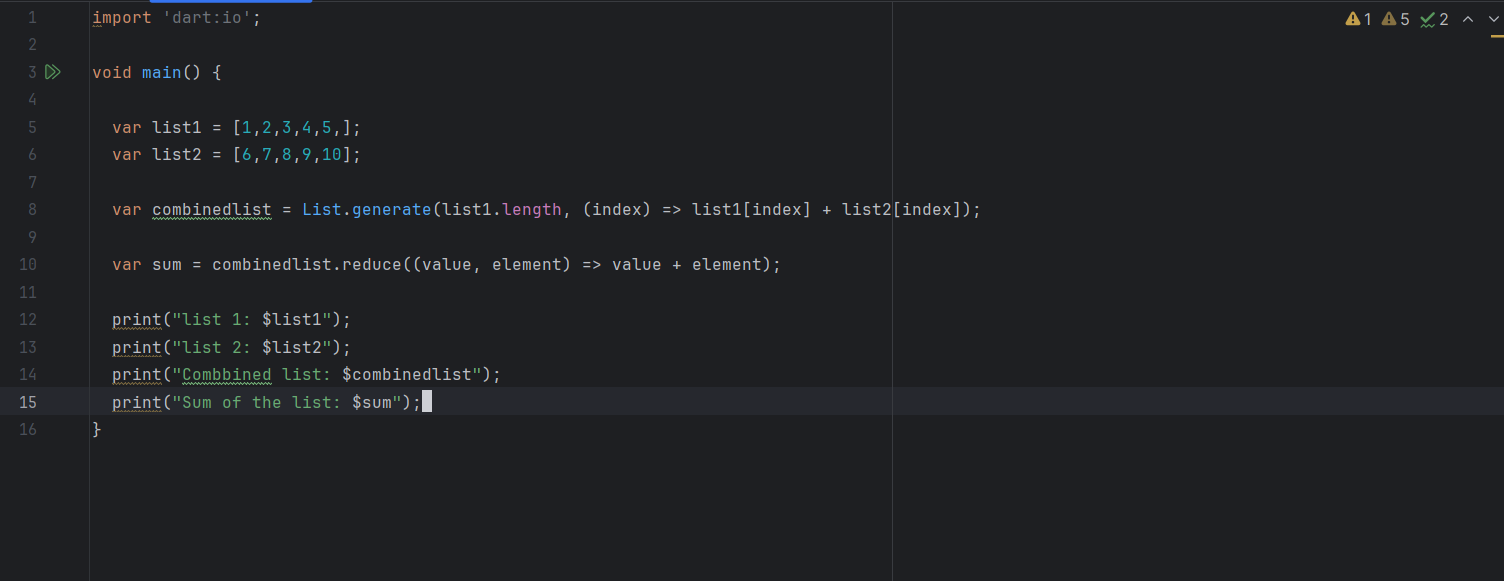
Output



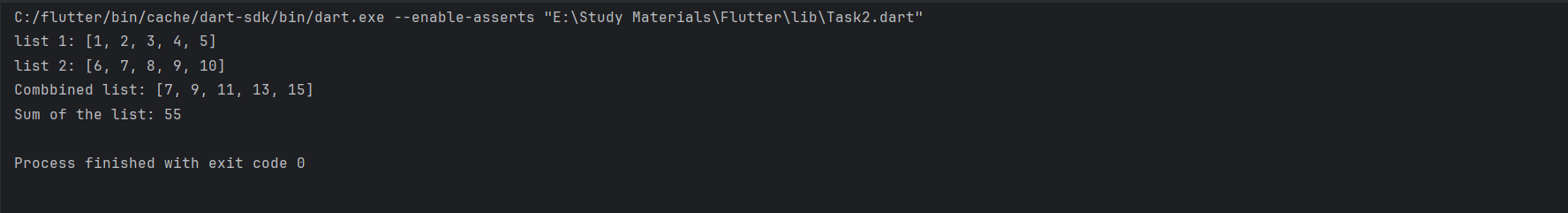
**Task 10: Combining and Reducing**

1. Create two lists of integers of equal length.

2. Combine the two lists element-wise by adding corresponding elements together. 3. Reduce the combined list to a single integer by calculating the sum of all elements.



Output

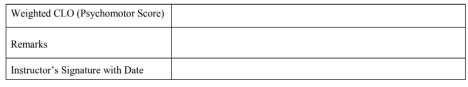
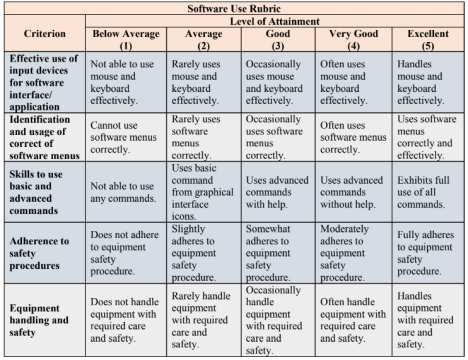


DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

**NED University of Engineering & Technology** 

**Department of Software Engineering**

**Course Code and Title: MOBILE APPLICATION DEVELOPMENT (SE-487)**

**Laboratory Session No. \_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**