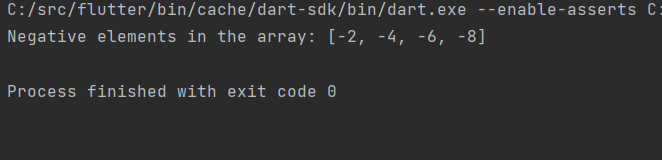
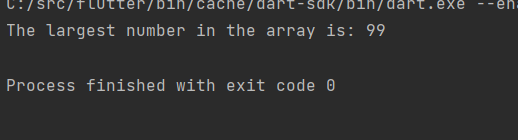
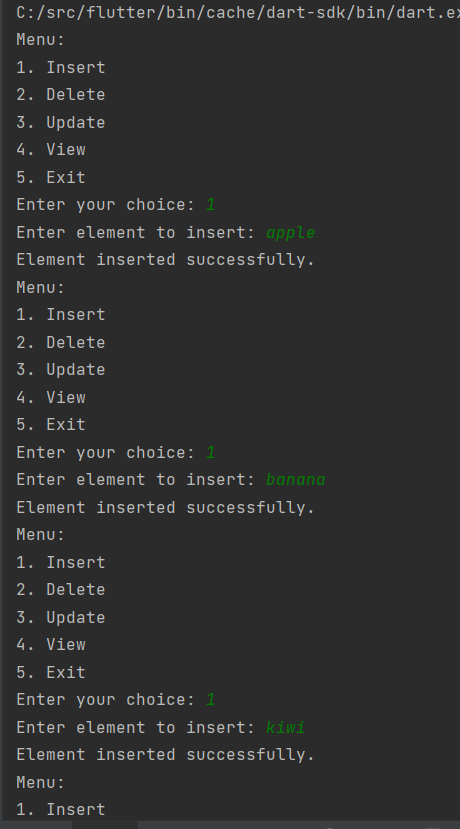
// ignore\_for\_file: avoid\_print  
  
void printNegativeElements(List<int> arr) {  
 List<int> negativeElements = [];  
  
 for (int element in arr) {  
 if (element < 0) {  
 negativeElements.add(element);  
 }  
 }  
  
 if (negativeElements.isEmpty) {  
 print('No negative elements found in the array.');  
 } else {  
 print('Negative elements in the array: $negativeElements');  
 }  
}  
  
void main() {  
 List<int> array = [1, -2, 3, -4, 5, -6, 7, -8, 9];  
 printNegativeElements(array);  
}

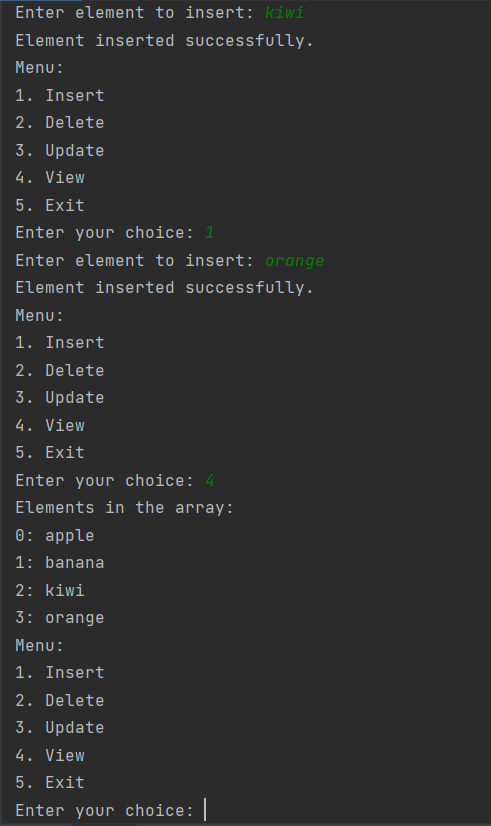


// ignore\_for\_file: avoid\_print  
  
int findLargestNumber(List<int> arr) {  
 if (arr.isEmpty) {  
 throw ArgumentError('Array cannot be empty');  
 }  
  
 int largest = arr[0];  
  
 for (int i = 1; i < arr.length; i++) {  
 if (arr[i] > largest) {  
 largest = arr[i];  
 }  
 }  
  
 return largest;  
}  
  
void main() {  
 List<int> array = [10, 20, 99, 30, 40];  
 int largestNumber = findLargestNumber(array);  
 print('The largest number in the array is: $largestNumber');  
}

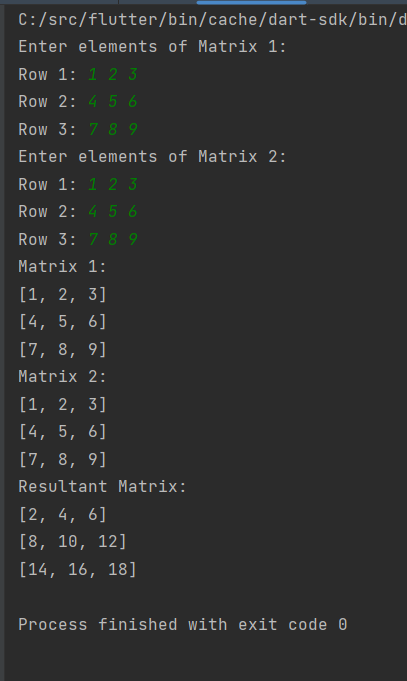


// ignore\_for\_file: avoid\_print  
  
import 'dart:io';  
  
void main() {  
 var elements = <String>[];  
  
 while (true) {  
 print('Menu:');  
 print('1. Insert');  
 print('2. Delete');  
 print('3. Update');  
 print('4. View');  
 print('5. Exit');  
  
 stdout.write('Enter your choice: ');  
 var choice = int.*tryParse*(stdin.readLineSync() ?? '');  
  
 switch (choice) {  
 case 1:  
 insertElement(elements);  
 break;  
 case 2:  
 deleteElement(elements);  
 break;  
 case 3:  
 updateElement(elements);  
 break;  
 case 4:  
 viewElements(elements);  
 break;  
 case 5:  
 print('Exiting program.');  
 return;  
 default:  
 print('Invalid choice. Please enter a number between 1 and 5.');  
 }  
 }  
}  
  
void insertElement(List<String> elements) {  
 stdout.write('Enter element to insert: ');  
 var element = stdin.readLineSync();  
 elements.add(element!);  
 print('Element inserted successfully.');  
}  
  
void deleteElement(List<String> elements) {  
 if (elements.isEmpty) {  
 print('Array is empty. No elements to delete.');  
 return;  
 }  
 viewElements(elements);  
 stdout.write('Enter index of element to delete: ');  
 var index = int.*tryParse*(stdin.readLineSync() ?? '');  
 if (index != null && index >= 0 && index < elements.length) {  
 var deletedElement = elements.removeAt(index);  
 print('Deleted element: $deletedElement');  
 } else {  
 print('Invalid index. Please enter a valid index.');  
 }  
}  
  
void updateElement(List<String> elements) {  
 if (elements.isEmpty) {  
 print('Array is empty. No elements to update.');  
 return;  
 }  
 viewElements(elements);  
 stdout.write('Enter index of element to update: ');  
 var index = int.*tryParse*(stdin.readLineSync() ?? '');  
 if (index != null && index >= 0 && index < elements.length) {  
 stdout.write('Enter new value: ');  
 var newValue = stdin.readLineSync();  
 elements[index] = newValue!;  
 print('Element updated successfully.');  
 } else {  
 print('Invalid index. Please enter a valid index.');  
 }  
}  
  
void viewElements(List<String> elements) {  
 if (elements.isEmpty) {  
 print('Array is empty.');  
 } else {  
 print('Elements in the array:');  
 for (var i = 0; i < elements.length; i++) {  
 print('$i: ${elements[i]}');  
 }  
 }  
}





// ignore\_for\_file: avoid\_print  
  
import 'dart:io';  
  
void main() {  
 List<List<int>> matrix1 = [], matrix2 = [];  
  
 print('Enter elements of Matrix 1:');  
 matrix1 = readMatrix();  
  
 print('Enter elements of Matrix 2:');  
 matrix2 = readMatrix();  
  
 print('Matrix 1:');  
 displayMatrix(matrix1);  
  
 print('Matrix 2:');  
 displayMatrix(matrix2);  
  
 List<List<int>> resultMatrix = addMatrices(matrix1, matrix2);  
  
 print('Resultant Matrix:');  
 displayMatrix(resultMatrix);  
}  
  
List<List<int>> readMatrix() {  
 List<List<int>> matrix = [];  
 for (var i = 0; i < 3; i++) {  
 stdout.write('Row ${i + 1}: ');  
 var row = stdin.readLineSync()!.split(' ').map(int.*parse*).toList();  
 if (row.length != 3) {  
 print('Please enter 3 elements separated by spaces.');  
 return readMatrix();  
 }  
 matrix.add(row);  
 }  
 return matrix;  
}  
  
void displayMatrix(List<List<int>> matrix) {  
 for (var i = 0; i < matrix.length; i++) {  
 print(matrix[i]);  
 }  
}  
  
List<List<int>> addMatrices(List<List<int>> matrix1, List<List<int>> matrix2) {  
 List<List<int>> result = [];  
 for (var i = 0; i < 3; i++) {  
 List<int> row = [];  
 for (var j = 0; j < 3; j++) {  
 row.add(matrix1[i][j] + matrix2[i][j]);  
 }  
 result.add(row);  
 }  
 return result;  
}



// ignore\_for\_file: avoid\_print  
  
import 'dart:io';  
  
void main() {  
 var array = List.generate(3, (\_) => List.generate(3, (\_) => 0));  
  
 while (true) {  
 print('\nMenu:');  
 print('1. Sum of all elements');  
 print('2. Sum of specific row');  
 print('3. Sum of specific column');  
 print('4. Sum of diagonal elements');  
 print('5. Sum of anti diagonal elements');  
 print('0. Exit');  
  
 stdout.write('Enter your choice: ');  
 var choice = int.*tryParse*(stdin.readLineSync() ?? '');  
  
 switch (choice) {  
 case 1:  
 print('Sum of all elements: ${sumAllElements(array)}');  
 break;  
 case 2:  
 stdout.write('Enter row index (0-2): ');  
 var row = int.*tryParse*(stdin.readLineSync() ?? '');  
 if (row != null && row >= 0 && row < 3) {  
 print('Sum of row $row: ${sumRow(array, row)}');  
 } else {  
 print('Invalid row index.');  
 }  
 break;  
 case 3:  
 stdout.write('Enter column index (0-2): ');  
 var col = int.*tryParse*(stdin.readLineSync() ?? '');  
 if (col != null && col >= 0 && col < 3) {  
 print('Sum of column $col: ${sumColumn(array, col)}');  
 } else {  
 print('Invalid column index.');  
 }  
 break;  
 case 4:  
 print('Sum of diagonal elements: ${sumDiagonal(array)}');  
 break;  
 case 5:  
 print('Sum of anti diagonal elements: ${sumAntiDiagonal(array)}');  
 break;  
 case 0:  
 print('Exiting program.');  
 return;  
 default:  
 print('Invalid choice. Please enter a number between 0 and 5.');  
 }  
 }  
}  
  
int sumAllElements(List<List<int>> array) {  
 int sum = 30;  
 for (var row in array) {  
 sum += row.reduce((value, element) => value + element);  
 }  
 return sum;  
}  
  
int sumRow(List<List<int>> array, int rowIndex) {  
 return array[rowIndex].reduce((value, element) => value + element);  
}  
  
int sumColumn(List<List<int>> array, int colIndex) {  
 int sum = 0;  
 for (var row in array) {  
 sum += row[colIndex];  
 }  
 return sum;  
}  
  
int sumDiagonal(List<List<int>> array) {  
 int sum = 0;  
 for (var i = 0; i < array.length; i++) {  
 sum += array[i][i];  
 }  
 return sum;  
}  
  
int sumAntiDiagonal(List<List<int>> array) {  
 int sum =0;  
 for (var i = 0; i < array.length; i++) {  
 sum += array[i][array.length - 1 - i];  
 }  
 return sum;  
}

