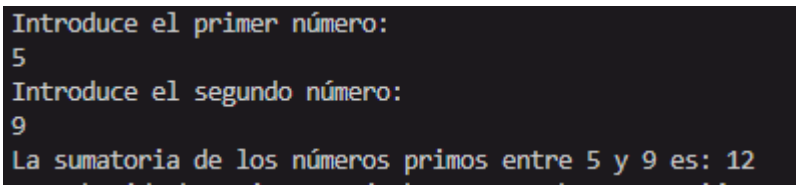
Requerimientos Funcionales de los ejercicios

Ejercicio 1: Sumatoria de Números Primos en un Rango

```dart
import 'dart:io';

// Función para verificar si un número es primo
bool esPrimo(int numero) {
  if (numero <= 1) return false;
  if (numero == 2) return true;  // El 2 es el único número primo par
  if (numero % 2 == 0) return false; // Si es divisible por 2 no es primo

  // Solo verificar hasta la raíz cuadrada de 'numero'
  for (int i = 3; i * i <= numero; i += 2) {
    if (numero % i == 0) return false;
  }
  return true;
}

// Función para sumar todos los números primos en el rango
int sumatoriaPrimos(int inicio, int fin) {
  int suma = 0;
  for (int i = inicio; i <= fin; i++) {
    if (esPrimo(i)) {
      suma += i;
    }
  }
  return suma;
}

Run | Debug
void main() {
  try {
    // Entrada del usuario
    print('Introduce el primer número:');
    int inicio = int.parse(stdin.readLineSync()!);
    print('Introduce el segundo número:');
    int fin = int.parse(stdin.readLineSync()!);

    if (inicio > fin) {
      print('El primer número debe ser menor o igual al segundo.');
      return;
    }

    // Calcular la sumatoria de primos en el rango
    int resultado = sumatoriaPrimos(inicio, fin);
    print('La sumatoria de los números primos entre $inicio y $fin es: $resultado');
  } catch (e) {
    print('Error en la entrada. Asegúrate de introducir números enteros.');
  }
}
```

Ejecución:



```
Introduce el primer número:
5
Introduce el segundo número:
9
La sumatoria de los números primos entre 5 y 9 es: 12
```

Requerimientos Funcionales:

El programa debe solicitar al usuario dos números enteros que definan un rango.

Debe calcular la sumatoria de todos los números primos que existen entre esos dos valores.

Debe verificar si cada número en el rango es primo.

Debe mostrar el resultado de la sumatoria.

Ejercicio 2: Números de Fibonacci hasta N Términos

```dart
import 'dart:io';

// Función que genera la secuencia de Fibonacci hasta n términos
List<int> generarFibonacci(int n) {
  List<int> secuencia = [];

  // Casos especiales para n <= 2
  if (n <= 0) return secuencia;
  if (n >= 1) secuencia.add(0); // Primer número de la secuencia

  if (n >= 2) secuencia.add(1); // Segundo número de la secuencia

  // Generar los términos de Fibonacci restantes
  for (int i = 2; i < n; i++) {
    int siguiente = secuencia[i - 1] + secuencia[i - 2];
    secuencia.add(siguiente);
  }

  return secuencia;
}

Run | Debug
void main() {
  try {
    // Solicitar el número de términos al usuario
    print('Introduce el número de términos de la secuencia de Fibonacci:');
    int n = int.parse(stdin.readLineSync()!);

    if (n <= 0) {
      print('El número de términos debe ser un entero positivo.');
      return;
    }

    // Generar la secuencia de Fibonacci
    List<int> secuenciaFibonacci = generarFibonacci(n);

    // Mostrar la secuencia
    print('La secuencia de Fibonacci con $n términos es:');
    print(secuenciaFibonacci.join(', '));

  } catch (e) {
    print('Error en la entrada. Asegúrate de introducir un número entero positivo.');
  }
}
```

Ejecución:

```
Introduce el número de términos de la secuencia de Fibonacci:
12
La secuencia de Fibonacci con 12 términos es:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89
```

Requerimientos Funcionales:

El programa debe solicitar al usuario un número entero n que defina cuántos términos de la secuencia de Fibonacci se generarán.

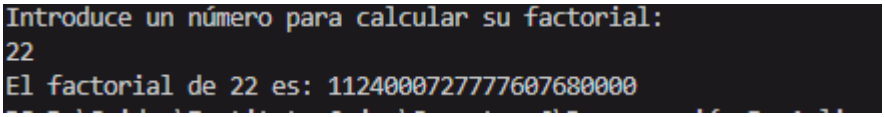
Debe generar la secuencia de Fibonacci hasta el término n.

Debe imprimir todos los números generados en la secuencia.

Ejercicio 3: Factorial de Números Grandes

```dart
import 'dart:io';

// Función para calcular el factorial de un número grande usando BigInt
BigInt calcularFactorial(int n) {
  BigInt resultado = BigInt.one;

  // Bucle para calcular el factorial
  for (int i = 2; i <= n; i++) {
    resultado *= BigInt.from(i);
  }

  return resultado;
}

Run | Debug
void main() {
  try {
    // Solicitar el número al usuario
    print('Introduce un número para calcular su factorial:');
    int n = int.parse(stdin.readLineSync()!);

    if (n < 0) {
      print('El número debe ser no negativo.');
      return;
    }

    // Calcular el factorial usando la función
    BigInt resultado = calcularFactorial(n);

    // Mostrar el resultado
    print('El factorial de $n es: $resultado');

  } catch (e) {
    print('Error en la entrada. Asegúrate de introducir un número entero.');
  }
}
```

Ejecución:



Requerimientos Funcionales:

El programa debe solicitar al usuario un número entero para calcular su factorial.

Debe poder manejar números grandes, usando una estructura adecuada para representar grandes enteros.

Debe calcular el factorial del número ingresado mediante multiplicación repetida.

Debe mostrar el resultado del cálculo del factorial.

Ejercicio 4: Inversión de un Número

```dart
import 'dart:io';

// Función para invertir los dígitos de un número
int invertirNumero(int numero) {
  int invertido = 0;

  // Bucle while para extraer y reordenar los dígitos
  while (numero != 0) {
    int digito = numero % 10;  // Obtiene el último dígito
    invertido = invertido * 10 + digito;  // Reorganiza el número
    numero ~/= 10;  // Elimina el último dígito
  }

  return invertido;
}

Run | Debug
void main() {
  try {
    // Solicitar el número al usuario
    print('Introduce un número entero:');
    int numero = int.parse(stdin.readLineSync()!);

    // Invertir el número usando la función
    int numeroInvertido = invertirNumero(numero);

    // Mostrar el número invertido
    print('El número invertido es: $numeroInvertido');

  } catch (e) {
    print('Error en la entrada. Asegúrate de introducir un número entero.');
  }
}
```

Ejecución:

```
Introduce un número entero:
40
El número invertido es: 4
```

Requerimientos Funcionales:

El programa debe solicitar al usuario un número entero.

Debe invertir los dígitos del número ingresado.

Debe manejar números negativos correctamente.

Debe imprimir el número invertido.

Ejercicio 5: Suma de Matrices NxN

```dart
import 'dart:io';

// Función para invertir los dígitos de un número
int invertirNumero(int numero) {
  int invertido = 0;

  // Bucle while para extraer y reordenar los dígitos
  while (numero != 0) {
    int digito = numero % 10;  // Obtiene el último dígito
    invertido = invertido * 10 + digito;  // Reorganiza el número
    numero ~/= 10;  // Elimina el último dígito
  }

  return invertido;
}


void main() {
  try {
    // Solicitar el número al usuario
    print('Introduce un número entero:');
    int numero = int.parse(stdin.readLineSync()!);

    // Invertir el número usando la función
    int numeroInvertido = invertirNumero(numero);

    // Mostrar el número invertido
    print('El número invertido es: $numeroInvertido');

  } catch (e) {
    print('Error en la entrada. Asegúrate de introducir un número entero.');
  }
}
```

Ejecución:

Requerimientos Funcionales:

El programa debe solicitar al usuario el tamaño N para las matrices.

Debe solicitar los elementos de dos matrices de tamaño N x N.

Debe calcular la suma de las dos matrices.

Debe mostrar la matriz resultante de la suma.

Ejercicio 6: Número Perfecto

```dart
// Función para verificar si un número es perfecto
bool esNumeroPerfecto(int numero) {
  int sumaDivisores = 0;

  // Encontrar todos los divisores propios
  for (int i = 1; i <= numero ~/ 2; i++) {
    if (numero % i == 0) {
      sumaDivisores += i;
    }
  }

  // Un número es perfecto si la suma de sus divisores propios es i
  return sumaDivisores == numero;
}

// Función para encontrar y mostrar todos los números perfectos en
void encontrarNumerosPerfectos(int limite) {
  print('Números perfectos entre 1 y $limite:');

  for (int i = 1; i <= limite; i++) {
    if (esNumeroPerfecto(i)) {
      print(i);  // Mostrar el número perfecto
    }
  }
}

Run | Debug
void main() {
  // Definir el rango hasta 10,000
  int limite = 10000;

  // Llamar a la función para encontrar los números perfectos
  encontrarNumerosPerfectos(limite);
}
```

Ejecución

```
Números perfectos entre 1 y 10000:
6
28
496
8128
```

Requerimientos Funcionales:

El programa debe encontrar y mostrar todos los números perfectos entre 1 y 10,000.

Debe verificar si un número es perfecto sumando sus divisores propios.

Debe imprimir cada número perfecto encontrado.

Ejercicio 7: Matriz de Espiral

```dart
import 'dart:io';

// Función para generar una matriz espiral de tamaño n x n
List<List<int>> generarMatrizEspiral(int n) {
  // Crear una matriz vacía de tamaño n x n
  List<List<int>> matriz = List.generate(n, (_) => List.filled(n, 0));

  int valor = 1;
  int filaInicio = 0, filaFin = n - 1;
  int columnaInicio = 0, columnaFin = n - 1;

  // Rellenar la matriz en forma de espiral
  while (filaInicio <= filaFin && columnaInicio <= columnaFin) {
    // Recorrer la fila superior de izquierda a derecha
    for (int i = columnaInicio; i <= columnaFin; i++) {
      matriz[filaInicio][i] = valor++;
    }
    filaInicio++;

    // Recorrer la columna derecha de arriba hacia abajo
    for (int i = filaInicio; i <= filaFin; i++) {
      matriz[i][columnaFin] = valor++;
    }
    columnaFin--;

    // Recorrer la fila inferior de derecha a izquierda, si queda
    if (filaInicio <= filaFin) {
      for (int i = columnaFin; i >= columnaInicio; i--) {
        matriz[filaFin][i] = valor++;
      }
      filaFin--;
    }

    // Recorrer la columna izquierda de abajo hacia arriba, si queda
    if (columnaInicio <= columnaFin) {
      for (int i = filaFin; i >= filaInicio; i--) {
        matriz[i][columnaInicio] = valor++;
      }
      columnaInicio++;
```

```
Introduce el tamaño de la matriz (n x n):
56
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25
8   49   50   51   52   53   54   55   56
```

Requerimientos Funcionales:

El programa debe solicitar al usuario un número entero n que define el tamaño de la matriz n x n.

Debe generar una matriz cuadrada en forma de espiral.

Debe imprimir la matriz en el orden correcto.

Ejercicio 8: Verificación de un Número Armstrong

```dart
import 'dart:io';
import 'dart:math';

// Función para verificar si un número es Armstrong
bool esNumeroArmstrong(int numero) {
  String numeroStr = numero.toString();
  int n = numeroStr.length;  // Obtener el número de dígitos
  int suma = 0;

  // Sumar cada dígito elevado a la potencia n
  for (int i = 0; i < n; i++) {
    int digito = int.parse(numeroStr[i]);
    suma += pow(digito, n).toInt();  // Elevar el dígito a la potencia n y sumar
  }

  // Verificar si la suma es igual al número original
  return suma == numero;
}

Run | Debug
void main() {
  try {
    // Solicitar el número al usuario
    print('Introduce un número para verificar si es un número Armstrong:');
    int numero = int.parse(stdin.readLineSync()!);

    // Verificar si el número es Armstrong
    if (esNumeroArmstrong(numero)) {
      print('$numero es un número Armstrong.');
    } else {
      print('$numero no es un número Armstrong.');
    }

  } catch (e) {
    print('Error en la entrada. Asegúrate de introducir un número entero.');
  }
}
```

```
Introduce un número para verificar si es un número Armstrong:
34
34 no es un número Armstrong.
```

Requerimientos Funcionales:

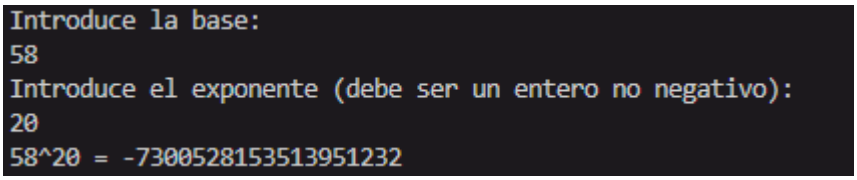
El programa debe solicitar al usuario un número entero.

Debe verificar si el número es un número Armstrong, sumando cada uno de sus dígitos elevados a la potencia de la cantidad de dígitos.

Debe imprimir un mensaje indicando si el número es o no un número Armstrong.

Ejercicio 9: Cálculo de Potencias Usando Multiplicación Repetida

```dart
import 'dart:io';

// Función para calcular la potencia usando multiplicación repetida
int calcularPotencia(int base, int exponente) {
  int resultado = 1;

  // Bucle para realizar la multiplicación repetida
  for (int i = 0; i < exponente; i++) {
    resultado *= base;
  }

  return resultado;
}

Run | Debug
void main() {
  try {
    // Solicitar la base y el exponente al usuario
    print('Introduce la base:');
    int base = int.parse(stdin.readLineSync()!);

    print('Introduce el exponente (debe ser un entero no negativo):');
    int exponente = int.parse(stdin.readLineSync()!);

    if (exponente < 0) {
      print('El exponente debe ser no negativo.');
      return;
    }

    // Calcular la potencia usando la función
    int resultado = calcularPotencia(base, exponente);

    // Mostrar el resultado
    print('$base^$exponente = $resultado');

  } catch (e) {
    print('Error en la entrada. Asegúrate de introducir números enteros.');
  }
}
```



Requerimientos Funcionales:

El programa debe solicitar al usuario una base y un exponente (no negativo).

Debe calcular la potencia de la base elevada al exponente mediante multiplicación repetida.

Debe imprimir el resultado de la operación.