**Ejercicio 1: Sumatoria de Números Primos en un Rango**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar dos números enteros, asegurando que el primero sea menor que el segundo.
2. **Cálculo de Números Primos**:
   * Verificar si cada número en el rango es primo.
3. **Sumatoria**:
   * Calcular la suma de los números primos en el rango.
4. **Salida de Resultados**:
   * Mostrar la sumatoria de los números primos.
5. **Validaciones**:
   * Asegurar que el primer número sea menor que el segundo y manejar entradas no válidas.

**Ejercicio 2: Números de Fibonacci hasta N términos**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar un número entero positivo n al usuario.
2. **Generación de la Secuencia**:
   * Utilizar un bucle for o while para calcular la secuencia de Fibonacci hasta el n-ésimo término.
3. **Almacenamiento de Resultados**:
   * Guardar los números de la secuencia en una lista.
4. **Salida de Resultados**:
   * Mostrar la secuencia de Fibonacci generada.
5. **Validaciones**:
   * Verificar que el número n ingresado sea positivo.

**Ejercicio 3: Factorial de Números Grandes**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar un número entero n al usuario.
2. **Cálculo del Factorial**:
   * Utilizar un bucle para calcular el factorial de n usando BigInt.
3. **Manejo de Números Grandes**:
   * Usar la clase BigInt para realizar cálculos con números grandes.
4. **Salida de Resultados**:
   * Mostrar el factorial de n calculado.
5. **Validaciones**:
   * Verificar que el número ingresado sea positivo.

**Ejercicio 4: Inversión de un Número**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar un número entero al usuario.
2. **Inversión de Dígitos**:
   * Utilizar un bucle while para extraer y reordenar los dígitos del número.
3. **Salida de Resultados**:
   * Mostrar el número invertido.
4. **Validaciones**:
   * Asegurar que la entrada sea válida y un número entero.

**Ejercicio 5: Suma de Matrices NxN**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar el tamaño N de las matrices al usuario.
2. **Entrada de Matrices**:
   * Solicitar los elementos de las dos matrices N x N.
3. **Suma de Matrices**:
   * Usar bucles for anidados para sumar las dos matrices.
4. **Salida de Resultados**:
   * Mostrar la matriz resultante de la suma.
5. **Validaciones**:
   * Asegurar que el tamaño N sea positivo y las matrices tengan las dimensiones correctas.

**Ejercicio 6: Número Perfecto**

**Requerimientos Funcionales:**

1. **Iteración de Rango**:
   * Iterar sobre el rango de números entre 1 y 10,000.
2. **Cálculo de Divisores Propios**:
   * Usar un bucle para sumar los divisores propios de cada número.
3. **Verificación de Perfección**:
   * Comprobar si la suma de los divisores es igual al número.
4. **Salida de Resultados**:
   * Mostrar todos los números perfectos encontrados.

**Ejercicio 7: Matriz de Espiral**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar el tamaño n para la matriz n x n.
2. **Construcción de la Matriz en Espiral**:
   * Usar bucles while y for para llenar la matriz en forma de espiral.
3. **Salida de Resultados**:
   * Mostrar la matriz en espiral generada.
4. **Validaciones**:
   * Asegurar que el tamaño n sea positivo.

**Ejercicio 8: Verificación de un Número Armstrong**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar un número entero n al usuario.
2. **Separación de Dígitos**:
   * Utilizar un bucle for para extraer y elevar cada dígito a la potencia correspondiente.
3. **Cálculo de Número Armstrong**:
   * Calcular la suma de los dígitos elevados a la potencia.
4. **Verificación**:
   * Comparar si la suma es igual al número original.
5. **Salida de Resultados**:
   * Mostrar si el número es un número Armstrong.

**Ejercicio 9: Cálculo de Potencias usando Multiplicación Repetida**

**Requerimientos Funcionales:**

1. **Entrada de Datos**:
   * Solicitar una **base** y un **exponente** enteros al usuario.
2. **Cálculo de Potencia**:
   * Usar un bucle while o for para calcular la potencia usando multiplicación repetida.
3. **Validaciones**:
   * Asegurar que el exponente sea mayor o igual a 0.
4. **Salida de Resultados**:
   * Mostrar el resultado de la potencia calculada.