Práctico 11: Aplicación de la Recursividad

Objetivo:

Comprender el uso de recursividad en problemas matemáticos simples.

Resultados de aprendizaje:

- ✓ Comprensión y aplicación de la recursividad: El estudiante será capaz de definir y comprender el concepto de recursividad, identificando los casos base y recursivos en una función recursiva.
- ✓ Diseño y desarrollo de algoritmos recursivos: El estudiante será capaz de diseñar funciones recursivas para resolver problemas complejos, descomponiendo el problema en subproblemas más sencillos.
- ✓ Resolución de problemas a través de la recursividad: El estudiante será capaz de aplicar la recursividad en la resolución de una variedad de problemas, como la búsqueda en estructuras de datos, el ordenamiento y la generación de estructuras combinatorias.

Ejercicios

1) Crea una función recursiva que calcule el factorial de un número. Luego, utiliza esa función para calcular y mostrar en pantalla el factorial de todos los números enteros entre 1 y el número que indique el usuario
2) Crea una función recursiva que calcule el valor de la serie de Fibonacci en la posición indicada. Posteriormente, muestra la serie completa hasta la posición que el usuario especifique.
3) Crea una función recursiva que calcule la potencia de un número base elevado a un exponente, utilizando la fórmula $n^m = n * n^{(m-1)}$. Prueba esta función en un algoritmo general.
4) Crear una función recursiva en Python que reciba un número entero positivo en base decimal y devuelva su representación en **binario** como una **cadena de texto**.

Cuando representamos un número en binario, lo expresamos usando solamente ceros (0) y unos (1), en base 2. Para convertir un número decimal a binario, se puede seguir este procedimiento:

1. Dividir el número por 2.
2. Guardar el resto (0 o 1).
3. Repetir el proceso con el cociente hasta que llegue a 0.
4. Los restos obtenidos, leídos de abajo hacia arriba, forman el número binario.

Ejemplo:

Convertir el número **10** a binario:

10 ÷ 2 = 5 resto: 0
5 ÷ 2 = 2 resto: 1
2 ÷ 2 = 1 resto: 0
1 ÷ 2 = 0 resto: 1

Leyendo los restos de abajo hacia arriba: 1 0 1 0 → El resultado binario es **"1010"**.

5) Implementá una función recursiva llamada es_palindromo(palabra) que reciba una cadena de texto sin espacios ni tildes, y devuelva True si es un palíndromo o False si no lo es.

   📌 Requisitos:
   La solución debe ser recursiva.
   No se debe usar [::-1] ni la función reversed().

6) Escribí una función recursiva en Python llamada suma_digitos(n) que reciba un número entero positivo y devuelva la suma de todos sus dígitos.

   📌 Restricciones:
   No se puede convertir el número a string.
   Usá operaciones matemáticas (%, //) y recursión.
   Ejemplos:
   suma_digitos(1234) → 10 (1 + 2 + 3 + 4)
   suma_digitos(9) → 9
   suma_digitos(305) → 8 (3 + 0 + 5)

7) Un niño está construyendo una pirámide con bloques. En el nivel más bajo coloca n bloques, en el siguiente nivel uno menos (n - 1), y así sucesivamente hasta llegar al último nivel con un solo bloque.

   Escribí una función recursiva contar_bloques(n) que reciba el número de bloques en el nivel más bajo y devuelva el total de bloques que necesita para construir toda la pirámide.

   💡 Ejemplos:
   contar_bloques(1) → 1 (1)
   contar_bloques(2) → 3 (2 + 1)
   contar_bloques(4) → 10 (4 + 3 + 2 + 1)

8) Escribí una función recursiva llamada contar_digito(numero, digito) que reciba un número entero positivo (numero) y un dígito (entre 0 y 9), y devuelva cuántas veces aparece ese dígito dentro del número.

   💡 Ejemplos:
   contar_digito(12233421, 2) → 3
   contar_digito(5555, 5) → 4

contar_digito(123456, 7) → 0