Resolución y explicación detallada de ejercicios sobre recursividad

1) Factorial de un número

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

Explicación paso a paso:

- Caso base: si `n` es 0 o 1, devuelve 1.
- Paso recursivo: multiplica `n` por `factorial(n - 1)`.

Ejemplo con n = 4:

```
factorial(4)
→ 4 * factorial(3)
→ 4 * (3 * factorial(2))
→ 4 * (3 * (2 * factorial(1)))
→ 4 * (3 * (2 * 1)) = 24
```

---

2) Serie de Fibonacci

```
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
```

Explicación paso a paso:

- Caso base: si `n` es 0 o 1, devuelve `n`.
- Paso recursivo: suma `fibonacci(n - 1)` y `fibonacci(n - 2)`.

Ejemplo con n = 5:

```
fibonacci(5)
→ fibonacci(4) + fibonacci(3)
→ (fibonacci(3) + fibonacci(2)) + (fibonacci(2) + fibonacci(1))
→ ... (se ramifica hasta llegar a los casos base)
Resultado: 5
```

---

3) Potencia recursiva

```
def potencia(base, exponente):
    if exponente == 0:
        return 1
    else:
        return base * potencia(base, exponente - 1)
```

Explicación paso a paso:

- Caso base: cualquier número elevado a 0 es 1.
- Paso recursivo: multiplica la base por el resultado de elevarla al exponente - 1.

Ejemplo con base = 2, exponente = 3:

```
potencia(2, 3)
→ 2 * potencia(2, 2)
→ 2 * (2 * potencia(2, 1))
→ 2 * (2 * (2 * potencia(2, 0)))
→ 2 * 2 * 2 * 1 = 8
```

---

4) Conversión de decimal a binario

```
def decimal_a_binario(n):
    if n == 0:
        return ""
    else:
        return decimal_a_binario(n // 2) + str(n % 2)
```

Explicación paso a paso:

- Caso base: si `n` es 0, se devuelve una cadena vacía.
- Paso recursivo: divide el número por 2 y concatena el resto al resultado de la llamada recursiva.

Ejemplo con n = 10:

```
decimal_a_binario(10)
→ decimal_a_binario(5) + '0'
→ (decimal_a_binario(2) + '1') + '0'
→ ((decimal_a_binario(1) + '0') + '1') + '0'
→ (((decimal_a_binario(0) + '1') + '0') + '1') + '0'
→ "" + '1' + '0' + '1' + '0' = "1010"
```

---

5) Verificar si una palabra es palíndromo

```
def es_palindromo(palabra):
    if len(palabra) <= 1:
        return True
    if palabra[0] != palabra[-1]:
        return False
    return es_palindromo(palabra[1:-1])
```

Explicación paso a paso:

- Caso base: si la longitud es 0 o 1, es palíndromo.
- Paso recursivo: compara el primer y último carácter. Si son iguales, se llama recursivamente con la subcadena central.

Ejemplo con palabra = "radar":

```
es_palindromo("radar")
→ 'r' == 'r', sigue con "ada"
→ 'a' == 'a', sigue con "d"
→ longitud <= 1 → True
```

---

6) Suma de dígitos de un número

```
def suma_digitos(n):
    if n < 10:
        return n
    else:
        return n % 10 + suma_digitos(n // 10)
```

Explicación paso a paso:

- Caso base: si el número es menor a 10, lo devuelve.
- Paso recursivo: extrae el último dígito (n % 10) y lo suma al resultado de aplicar la función al número sin ese dígito.

Ejemplo con n = 305:

```
suma_digitos(305)
→ 5 + suma_digitos(30)
→ 5 + (0 + suma_digitos(3))
→ 5 + 0 + 3 = 8
```

---

7) Contar bloques en una pirámide

```
def contar_bloques(n):
    if n == 1:
        return 1
    else:
        return n + contar_bloques(n - 1)
```

Explicación paso a paso:

- Caso base: si hay un solo nivel, devuelve 1.
- Paso recursivo: suma la cantidad de bloques del nivel actual con los del nivel anterior.

Ejemplo con n = 4:

```
contar_bloques(4)
→ 4 + contar_bloques(3)
→ 4 + (3 + contar_bloques(2))
→ 4 + (3 + (2 + contar_bloques(1)))
→ 4 + 3 + 2 + 1 = 10
```

---

8) Contar apariciones de un dígito en un número

```
def contar_digito(numero, digito):
    if numero == 0:
        return 0
    else:
```

```
        ultimo = numero % 10
        return (1 if ultimo == digito else 0) + contar_digito(numero
// 10, digito)
```

Explicación paso a paso:

- Caso base: si el número es 0, devuelve 0.
- Paso recursivo: compara el último dígito con el buscado, suma 1 si coincide, y continúa con el resto del número.

Ejemplo con numero = 12233421, digito = 2:

```
contar_digito(12233421, 2)
→ 1 (último no es 2) + contar_digito(1223342, 2)
→ 1 (último es 2) + contar_digito(122334, 2)
→ 0 + contar_digito(12233, 2)
→ 0 + contar_digito(1223, 2)
→ 0 + contar_digito(122, 2)
→ 1 + contar_digito(12, 2)
→ 1 + contar_digito(1, 2)
→ 0
→ Total: 3 apariciones
```