Recursividad

Temas incluidos en la guía

- Recursividad

Ejercicios

> Nota: (★☆☆, ★★☆, ★★★) Esta notación indica la dificultad (ascendente).

1. ★☆☆ Para cada inciso, implementá una función **recursiva** que reciba un número *z* y un entero *k* y retorne:

    1. `z + k` (sumando unos),
    2. `z * k` (sumas sucesivas), y
    3. `z ^ k` (multiplicaciones sucesivas).

2. ★☆☆ Implementar una función **recursiva** que retorne la cantidad de dígitos de un número entero.

3. ★☆☆ Implementar una función **recursiva** que reciba un entero no negativo e retorne dicho número espejado, es decir, para el 1234 retorna 4321.

4. ★☆☆ Escribir una función **recursiva** que calcule el n-ésimo número triangular (el número triangular de `n` es `1 + 2 + 3 + ··· + n`).

5. ★★☆ Implementar una función **recursiva** que dados dos números `n` y `b` retorne `True` si `n` es potencia de `b`.

6. ★★☆ Escribir una función **recursiva** que encuentre el mayor elemento de un vector.

7. ★★☆ Escribir una función **recursiva** que calcule el n-ésimo número de la sucesión de Fibonacci. La misma se define como `F_n = F_{n-1} + F_{n-2}, F_0 = F_1 = 1`.

    Dibuje un esquema (árbol de recursividad) de las llamadas recursivas para `n = 6`.

8. ★★☆ Implementar una función recursiva que calcule el máximo común divisor (gcd) de 2 enteros `x` e `y`. El gcd de los enteros `x` e `y` es el mayor entero que divide a ambos números con resto cero.

    - Resuelva primero por fuerza bruta: comience a probar números desde `y` en orden descendiente.

    - Una mejora es utilizar la siguiente definición recursiva:

        - si `y` es igual a 0, entonces `gcd(x, y)` es `x`
        - si no `gcd(x, y)` es `gcd(y, x % y)`.

    - Un tercer algoritmo es:

        ```
        si m y n son iguales:
            m es el gcd
        si m es mayor a n:
        ```

```
        el gcd(m, n) es gcd(m - n, n)
    si no:
        el gcd(m, n) es gcd(m, n - m)
```

9. ★★☆ Escribir una función recursiva que dado un arreglo de números, retorne el máximo, usando dividir y conquistar.

10. ★★☆ El triángulo de Pascal es un arreglo triangular de números que se define de la siguiente manera:

    - Las filas se enumeran desde `n = 0`, de arriba hacia abajo.
    - Los valores de cada fila se enumeran desde `k = 0` (de izquierda a derecha).
    - Los valores que se encuentran en los bordes del triángulo son 1.
    - Cualquier otro valor se calcula sumando los dos valores contiguos de la fila de arriba.

```
                               k=0
n=0                         1       k=1
n=1                     1       1       k=2
n=2                 1       2       1       k=3
n=3             1       3       3       1       k=4
n=4         1       4       6       4       1       k=5
n=5     1       5       10      10      5       1
n=6 1       6       15      20      15      6       1
k=0     k=1     k=2     k=3     k=4     k=5     k=6
```

    1. Escribir una función **recursiva** `pascal(n, k)` que calcule el número correspondiente. Tener en cuenta que:

        - `pascal(n, 0) = pascal(n, n) = 1` si `n >= 0`
        - `pascal(n, k) = pascal(n - 1, k) + pascal(n - 1, k - 1)` si `n > k > 0`.

    2. Dibujar el árbol de recursividad para alguna llamada de interés.

11. ★★☆ Escribir una función recursiva que aplique las reglas del *game* de tenis, hasta ganar el game. Si consideramos los puntajes de tenis como 0, 15, 30, 40, 50 (Ventaja), 60 (Punto), un jugador gana el *game* siempre que alcanza los 60 puntos. Si un jugador A con 40 puntos anota, y el otro jugador B tiene menos de 40 puntos, entonces A pasa directamente a 60 puntos y gana el game. Si, en cambio, ambos jugadores tienen 40 puntos, quien anota pasa a tener 50 puntos, o ventaja. Cuando un jugador tiene ventaja, puede ocurrir que gane el siguiente tanto, en cuyo caso gana el *game*, o puede ocurrir que el oponente gane el tanto, entonces pasan a tener ambos 40 puntos nuevamente.

12. ★★☆ Escribir una especificación apropiada para la siguiente función. Se asume que n es un número entero:

```python
def f(n):
    s = str(n)

    if len(s) <= 1:
        return s
```

```python
    return s[-1] + f(int(s[:-1]))
```

13. ★★☆ Dadas las siguientes funciones:

```python
def g(x):
    return h(str(x))

def h(x):
    if len(xs) == 1:
        return int(xs)

    n = int(xs[0]) + int(xs[1])

    if len(xs) == 2:
        return n
    else:
        return n + h(xs[2:])
```

   - ¿Qué retorna `g(2112)`?
   - Escriba una especificación para la función `h()`.