Casos de Estudio

Analice, diseñe y codifique los siguientes enunciados en Python

1. Diseñe la función recursiva pertenece(elem, lista) que retorne True si elem está en la lista y False en caso contrario. Se propone el siguiente análisis:

   Caso base: Si la lista está vacía, entonces retorna False
   Recursión: Si elem es el elemento que está al principio de lista entonces retorna True, en caso contrario retorna pertenece(elem, lista[1:]) sin el primer elemento

2. Diseñe la función recursiva potencia(b, n) donde b es la base y n es el exponente de la potencia $\mathbf{b}^n$. Se propone el siguiente análisis, utilizando las propiedades de las potencias de igual base:

   Caso base: retorna 1 si el exponente n es igual a 0
   Recursión: retorna $\mathbf{b}^{(n/2)} * \mathbf{b}^{(n/2)}$ si el exponente n es par, o sino
   retorna $\mathbf{b}^{(n-1)/2} * \mathbf{b}^{(n-1)/2} * \mathbf{b}$ si el exponente n es impar

   Nota: Debe utilizar una variable auxiliar, para el cálculo y retorno de la potencia, y hacer uso de la propiedad de las potencias de igual base, que es una de las ventajas principales de esta implementación. Se aprovecha el resultado calculado en lugar de tener que calcularlo dos veces, entonces la solución recursiva será más eficiente que la solución iterativa ¿Puede ud. comprobarlo?

Ejercicios

1. Analizar, realizar la prueba de escritorio para los valores de entrada solicitados y mostrar la salida de los siguientes códigos:

x = 345	n10 = 2835 , foranea = 16

def sd(n):
 if n == 0:
  return 0
 else:
  return n % 10 + sd(int(n / 10))
 	
x = int(input('x:'))
print(sd(x))
	def acad(n, b):
	  tcon = "0123456789ABCDEF"
	  if n < b:
	  return tcon[n]
	  else:
	  return acad(n // b, b) + tcon[n % b]
	
	n10 = int(input('Nro:'))
	foranea = int(input('Foránea:'))
	print(acad(n10,foranea))

n = 7091	nro = 584

def modRec(n):
  s = str(n)
  if len(s) <=1:
  return s
  return s[-1] + modRec(int(s[:-1]))
 	
n = int(input('nro:'))
print(modRec(n))
	def misterio(n):
	  if n < 10:
	  return (10 * n) + n
	  else:
	  a = misterio(n // 10)
	  b = misterio(n % 10)
	  return (100 * a) + b
	
	nro = int(input('nro:'))
	print(misterio(nro))

2. El siguiente código simula el juego de adivinar un número entre 0 y 10. Juega el usuario contra el random. Analice el código y responda las preguntas al final:

```
import random
def jugar(guess, intento):
    nro = int(input("Ingrese número: "))
    if guess != nro:
        if intento < 3:
            print("Fallaste")
            intento += 1
            jugar(guess, intento)
        else:
            print("El número es: ", guess)
    else:
        print("Bien! ganaste")
#Principal
guess = random.randint(0,10)
intento = 1
print("Adivine un número entre 0 y 10")
jugar(guess, intento)
```

- Identifique el bloque base y el bloque recursivo.
- Modifique el código de forma que muestre los intentos restantes.
- Realice la traza del algoritmo
- Identifique el comportamiento de los parámetros de la función jugar() en cada llamado.
- ¿Cuál es la condición de salida?

3. Diseñar una función recursiva que devuelva la sumatoria 1 + 2 + 3 + ... + n

4. Diseñar una función recursiva que devuelva el mayor elemento de una lista

5. Diseñar una función recursiva que imprima los elementos de una lista

6. Diseñar una función iterativa que devuelva una lista conteniendo los n primeros términos de la sucesión de Fibonacci. ¿Cómo mejoraría el código?

7. Un palíndromo es una cadena que es igual hacia adelante y hacia atrás. Por ejemplo, ‘aba’, ‘abba’, ‘abcba’, ‘aaa’, ‘ababa’, etc. Escriba una función recursiva es_palindrome(cadena) que devuelve True si la cadena es un palíndromo y False en caso contrario.

8. Escribe una función recursiva t_pascal(n) dónde n es un entero positivo y la función devuelve la n-ésima fila del triángulo de Pascal. Ej: print(t_pascal(8)) devuelve [1, 7, 21, 35, 35, 21, 7, 1]

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    1   6   15  20  15  6   1
  1   7   21  35  35  21  7   1
```