Estructuras Discretas 2022-1
Práctica 3: Recursión

Facultad de Ciencias, UNAM

Fecha de entrega: 19 de noviembre del 2021

Resuelva los siguientes ejercicios de manera **individual**. En el archivo `practica3.py` se encuentra la firma correspondiente para cada función donde tiene que completar con código. Después de haber resuelto todo, puede ejecutar dentro de la carpeta **ED-P3**, el comando `python3 -m unittest -v test_practica3.py` para asegurarse que la salida de cada función es correcta. Es importante mencionar que esto último no asegura que esté bien implementado, sin embargo podrá darse cuenta si su implementación es incorrecta.

En esta práctica es **obligatorio** el uso de recursión en todos los ejercicios y posibles funciones auxiliares.

1. Escriba la función `encripta_cadena` que se encarga de poner en distinto orden la misma, de tal manera que la cadena resultante sea difícil de leer pero fácil de convertirla a como estaba originalmente. Inicialmente cuenta con la cadena vacía, a esta va agregando los caracteres como se sigue:

   - Agrega el caracter que se encuentra en medio. Si el tamaño de la cadena es par, entonces se tomará el caracter que está a la izquierda de los dos centrales.
   - Agrega la cadena encriptada correspondiente a los caracteres de la izquierda hasta el caracter de en medio
   - Agrega la cadena encriptada correspondiente a la subcadena que empieza después del caracter de en medio y termina con el caracter que se encuentra más a la derecha.

   Ejemplo:

   ```
   >>> encripta_cadena("abc")
   "bac"
   >>> encripta_cadena("abcd")
   "bacd"
   >>> encripta_cadena("abcxcba")
   "xbacbca"
   ```

2. Escriba el código para la función `es_primo` que, dado un número cualquiera determine si es primo o no.

   Ejemplo:

   ```
   >>> es_primo(7)
   True
   >>> es_primo(62)
   False
   ```

3. Defina la función `decimal_binario` que dado un número en decimal devuelva su conversión a binario.

   Ejemplo:

   ```
   >>> decimal_binario(8)
   1000
   >>> decimal_binario(2)
   10
   >>> decimal_binario(21)
   10101
   ```

4. Defina la función `suma_triangulo` que dada una lista de números, regresa una lista de listas, donde la última es la lista original y representa el primer nivel. En los siguientes niveles se tiene una lista que tiene por elementos la suma de cada par de elementos consecutivos en el nivel anterior.

   Ejemplo:

```
>>> suma_triangulo([1,6,10])
    [[23],[7,16],[1,6,10]]
>>> suma_triangulo([2,4,6,8,10])
    [[96],[40,56],[16,24,32],[6,10,14,18],[2,4,6,8,10]]
```

5. Defina la función `suma_consecutivos` que dada una lista de números, devuelve una lista con la suma de cada par de número consecutivos.

   Ejemplo:

```
>>> suma_consecutivos([])
    []
>>> suma_consecutivos([13])
    [13]
>>> suma_consecutivos([1,2,3,4,5,6,7])
    [3,5,7,9,11,13]
```

6. Defina la función `suma_potencias` que dado un número y una potencia, devuelva la suma de de los dígitos al ser elevados a la potencia recibida.

   Ejemplo:

```
>>> suma_potencias(1235, 2)
# 1^2 + 2^2 + 3^2 + 5 ^2 = 1 + 4 + 9 + 25 = 39
39
```

7. Defina la función `oculta_caracter` que dado un caracter cualquiera, lo oculta cierto número de veces.

```
>>> oculta_caracter("x", 1)
"(x)"
>>> oculta_caracter("x", 3)
"(((x)))"
```

8. Una persona puede subir los escalones subiendo de uno en uno, subiendo de dos en dos o subiendo de tres en tres. Escriba la función `numero_de_formas` que determine de cuántas maneras esa persona puede subir los n escalones.

   Ejemplo:

```
>>>numero_de_formas(5)
13
>>>numero_de_formas(6)
24
>>>numero_de_formas(1)
1
```

9. Defina la función `consonante_a_f` que dada una cadena devuelve la cadena reemplazando las consonantes por f.

   Ejemplo:

```
>>> consonante_a_f("aeiou")
    aeiou
>>> consonante_a_f("consonante")
    foffofaffe
```