Instituto Tecnológico de Costa Rica

Escuela de Computación

Introducción a la Programación / Taller de programación.

Profesor: Ing. Jeff Schmidt Peralta

EJERCICIOS SOBRE PROGRAMACIÓN RECURSIVA PARA 1er PARCIAL.

1. Escriba una función recursiva `separa(num)` que reciba un número entero que debe ser de dos o más dígitos y retorne una tupla compuesta por dos números, los cuales se formarán intercalando las posiciones que tienen los dígitos. El primer número contiene los dígitos que ocupan una posición par, o sea los dígitos de **derecha a izquierda** en las posiciones 0, 2, 4, …, y el segundo número se formará con los dígitos que ocupan un lugar impar.

```
>>> separa(1234567)
(1357, 246)
>>> separa(1)
Error
```

```
>>> separa(13)
(3, 1)
>>> separa(987551)
(851, 975)
```

2. Escriba una función Python `clasifique(num)` que determine si el número dado es deficiente, perfecto o abundante. Se dice que un número entero positivo es perfecto si la suma de sus divisores exactos (excepto él mismo) es igual a dicho número, si la suma de sus divisores exactos es menor se dice que es deficiente y si esa suma de divisores es mayor entonces es abundante.

```
>>> clasifique(28)
'Perfecto'
```

```
>>> clasifique(30)
'Abundante'
```

3. En una competencia de talentos 10 jueces califican a los competidores en una escala de 1 a 100. Para obtener la calificación de un competidor, se eliminan la más alta y la más baja y se hace un promedio de las calificaciones restantes. Debe implementar usando recursividad la función Python calificacion(lista) que recibe una lista con las calificaciones de los jueces y que obtenga la calificación de un competidor.

```
>>> calificacion([6, 8, 8, 8, 8, 8, 8, 10, 8, 8])
8
```

4. Escriba una función parejas(num) en la cual num es un número entero y que cuente las veces que aparecen dígitos en parejas. Ejemplos:

```
>>> parejas(41234426601)        >>> parejas(333)
2                               1
>>> parejas(234601)             >>> parejas(9991222)
0                               2
```

5. Escriba una función a llamar `eliminar(lista, ele)` que reciba una lista y un elemento y elimine la primera aparición de ese elemento en la lista.

```
>>> eliminar([0, 5, 1, 2, 5, 3, 4], 5)
[0, 1, 2, 5, 3, 4]
>>> eliminar([0, 5, 1, 2, 5, 3, 4], 8)
[0, 5, 1, 2, 5, 3, 4]
```

6. Escriba una función a llamar `eliminar_todos(lista, ele)` que reciba una lista y un elemento y elimine todas las apariciones de ese elemento en la lista.

```
>>> eliminar_todos([0, 5, 1, 2, 5, 3, 4], 5)
[0, 1, 2, 3, 4]
>>> eliminar_todos ([0, 5, 1, 2, 5, 3, 4], 8)
[0, 5, 1, 2, 5, 3, 4]
```

7. Escriba una función recursiva a llamar `frente2`, que reciba una lista de números y devuelva la lista de todos sus elementos exceptuando los dos últimos. Si la lista tiene dos o menos elementos, debe retornar la lista nula.

```
>>> frente2([0, 1, 2, 3, 4])
[0, 1, 2]
```

8. Hacer una función llamada `invierta(lista)`, que reciba una lista e invierta el orden de sus elementos (sin utilizar la función reverse de Python):

```
>>> invierta([0, 1, 2, 3, 4])
[4, 3, 2, 1, 0]
```