

ESTUDIANTE: \_\_\_\_\_ CARNÉ: \_\_\_\_\_

## INSTRUCCIONES:

- El desarrollo es en Python 3 con material visto a la fecha. No se permite el uso de estos tipos de datos: secuencias (strings, listas, tuplas), diccionarios y conjuntos de Python.
- Haga un solo programa fuente con el nombre **práctica3\_su\_nombre.py** que contenga las funciones con los nombres indicados en cada ejercicio.
- Tiempo disponible para hacer el examen: 3 horas lectivas. Administre su tiempo, considere que la hora máxima para enviar la solución es: \_\_\_\_ .
- No se permite usar material de consulta: manual, impreso, digital, Internet, etc.
- Se permite usar Internet únicamente cuando termine la práctica para enviar la solución al tecDigital (DOCUMENTOS / PRÁCTICAS).
- En caso de fraudes se aplica el reglamento institucional.

## RECOMENDACIONES:

- Lea cuidadosamente los problemas que tiene que resolver (primer paso de la metodología de solución de problemas: entender el problema).
- Haga un esquema general de la solución (estudie el comportamiento del algoritmo) y luego proceda con su desarrollo en la computadora.
- Pruebe las funciones desde el modo comando.

**Ejercicio 1.** Desarrolle la función **buscar\_y\_reemplazar**. Recibe tres valores enteros:

- El dígito a buscar (número natural entre 0 y 9)
- Un número analizado (número natural)
- Un número de reemplazo (número natural entre 10 y 99)

Cada vez que aparezca el dígito a buscar en el número analizado, ese dígito debe ser reemplazado por el número de reemplazo. Forme y retorne un número que cumpla con este requerimiento. Valide restricciones: tipos de datos y valores de los datos de entrada, si hay errores retorne el mensaje respectivo. Ejemplos del funcionamiento:

```
>>> buscar_y_reemplazar(4, 12456405, 18)    → 1218561805
```

```
>>> buscar_y_reemplazar(0, 508000, 67)      → 5678676767
```

```
>>> buscar_y_reemplazar(9, 18117, 30)       → 18117
```

```
>>> buscar_y_reemplazar ("9", 18117, 13)
```

```
"ERROR: EL DIGITO A BUSCAR DEBE SER UN NÚMERO NATURAL ENTRE 0 Y 9"
```

```
>>> buscar_y_reemplazar (9, -1811, 13)
```

```
"ERROR: EL NÚMERO ANALIZADO DEBE SER UN NÚMERO NATURAL"
```

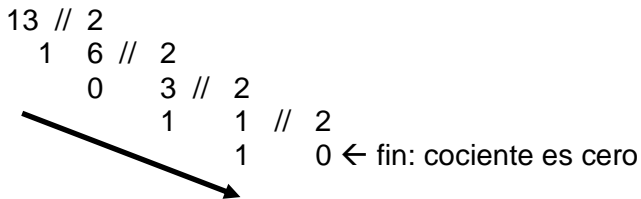
```
>>> buscar_y_reemplazar (9, 1811, 125)
```

```
"ERROR: EL NÚMERO DE REEMPLAZO DEBE SER UN NÚMERO NATURAL ENTRE 10 Y 99"
```

**Ejercicio 2.** Haga la función **convertir\_b10\_b2** que reciba un enteros en base decimal y retorne su conversión a base binaria (base 2 tiene 2 dígitos: 0, 1) según el ejemplo de funcionamiento.

Para hacer la conversión de base 10 a base 2 se usará el método de divisiones sucesivas que consiste en realizar divisiones del número decimal entre 2 (base binaria) hasta que el cociente sea cero. El primer residuo se convierte en el dígito menos significativo del resultado (el de más a la derecha) y el último residuo en el más significativo del número convertido. Por ejemplo el número decimal 13 se convierte en el número binario (base 2) 1101 de la siguiente forma:

```
13 // 2
  1  6 // 2
    0  3 // 2
      1  1 // 2
        1  0 ← fin: cociente es cero
```



No haga las validaciones de las restricciones, los datos vendrán según las especificaciones.

Ejemplos del funcionamiento:

```
>>> convertir_b10_b2(13)      → 1101
>>> convertir_b10_b2(428)     → 110101100
>>> convertir_b10_b2(4)       → 100
>>> convertir_b10_b2(0)       → 0
```

**Ejercicio 3.** En matemáticas la sucesión de Fibonacci es la siguiente sucesión infinita de números naturales:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Número de término: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...

Tal como se puede observar el término número 1 de la sucesión es el valor 0, el término número 2 es el valor 1, el término número 3 también es el valor 1, el término número 4 es el valor 2, etc.

En general el término número  $n$  (término  $n$ -ésimo) de esta sucesión es la suma los dos términos anteriores: término  $n-1$  + término  $n-2$ .

Desarrolle la función **fibonacci** que calcule el término número  $n$  de esta sucesión. Recibe el  $n$  y retorna el término respectivo. Valide que la entrada sea un número natural, si hay errores retorne el mensaje respectivo. Ejemplos del funcionamiento:

```
>>> fibonacci(8)
13 # el término número 8 de la sucesión tiene el valor 13: suma el valor del término 7 (8) y con el valor del término 6 (5)
```

```
>>> fibonacci(5)
3
Última línea
```