Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC1103 Introducción a la Programación

Ejercicios Propuestos

Rodrigo Toro Icarte

Introducción

El objetivo de esta guía es preparar el midterm del sábado. La solución de los ejercicios son cortas, pero requieren pensarlas bien.

Ejercicios

1. Programa una función que reciba una matriz y retorne la matriz rotada 90 grados en sentido horario. Ambas matrices deben estar codificadas como listas de litas. Por ejemplo, considera la matriz de la figura 1a. Al rotarla deberías obtener la figura 1b. Si continúas rotando deberías obtener la figura 1c, 1d y finalmente 1e, que es la matriz original.

1	2	3
4	5	6

(a)

4	1
5	2
6	3

(b)

6	5	4
3	2	1

(c)

3	6
2	5
1	4

(d)

1	2	3
4	5	6

(e)

Figura 1

Para probar tu función te recomiendo usar el siguiente código:

```
m = [[1,2,3],[4,5,6]]
for i in range(4):
  m = rotar(m) # <- tú debes implementar esta función
  print(m)
```

2. Considera un string `s` formado exclusivamente por ceros y unos. Queremos saber cuántas veces aparecen `k` unos consecutivos en `s`. Para ello implemente la función `contar(s,k)` que retorne el valor antes mencionado. Por ejemplo, con `k = 2` y `s = "01110"` la respuesta correcta es 2, porque el primer `11` es `"01110"`, y el segundo es `"01110"`. Notar que la función `count` no sirve para estos casos, porque `"01110".count("11")` es `1`.

3. Las matrices 2D se suelen codificar como listas de listas. Sin embargo, existe una codificación alternativa, que consiste en una tupla `(N,M,L)`, donde `N` es el número de filas, `M` es el número de columnas, y `L` es una lista con los elementos de la matriz. Por ejemplo, la matriz de la figura 1a quedaría codificada como `(2,3,[1,2,3,4,5,6])`. Implemente:

   a) `convertir_formato_tuplas(m)`: Que recibe una matriz codificada como lista de listas, y retorna la misma matriz en formato tupla. Por ejemplo: `convertir_formato_tuplas([[1,2,3],[4,5,6]])` debería retornar `(2,3,[1,2,3,4,5,6])`.

b) `convertir_formato_listas(m)`: Que recibe una matriz codificada como tupla y retorna la misma matriz codificada como lista de listas. Por ejemplo `convertir_formato_listas((2,3,[1,2,3,4,5,6]))` debería retornar `[[1,2,3],[4,5,6]]`.

4. Implementa la función `r_shift(n,k)` (con `n` y `k` enteros), que desplaza `k` veces el último dígito de `n` al inicio de este. Por ejemplo: `r_shift(1234,1)` → `2341`, `r_shift(1234,3)` → `4123`, `r_shift(1234,4)` → `1234`, y `r_shift(8,10)` → `8`. **No puede usar strings** para resolver este problema.

5. Considera la siguiente información:

   - El 1 de enero de 1900 fue lunes.
   - Los meses de Septiembre, Abril, Junio y Noviembre tienen 30 días. El resto tienen 31, salvo febrero que tiene 28 o 29 dependiendo si el año bisiesto.
   - Todo años es bisiesto si y solo si cumple alguna de las siguientes propiedades:
     - Es divisible por 4 y no por 100.
     - Es divisible por 400.

   Realiza un programa que cuente cuántos domingos han caído el primer día de un mes desde el 1/1/1900 hasta el 1/1/2000.[1]

6. ¿De cuántas maneras distintas puedo dar $1000 de vuelto utilizando monedas de $100, $50, $10 y $1?[2]

7. Los números triangulares, pentagonales y hexagonales se generan usando las siguientes fórmulas:

   - Triangulares: $T_n = n(n+1)/2$: 1, 3, 6, 10, 15...
   - Pentagonales: $P_n = n(3n-1)/2$: 1, 5, 12, 22, 35, ...
   - Hexagonales: $H_n = n(2n-1)$: 1, 6, 15, 28, 45, ...

   Es posible verificar que $T_{285} = P_{165} = H_{143} = 40755$. Busque el próximo número triangular que también sea pentagonal y hexagonal a la vez.[3]

8. **[SOLO PARA VALIENTES]** Considera un tablero de ajedrez con cierto número de fichas en él, denominadas superreinas. Las superreinas se pueden mover en forma horizontal, vertical y diagonal todas las casillas que deseen hasta toparse con otra pieza. Si una superreina al desplazarse se *sale* del tablero, aparecerá en el extremo opuesto del mismo. Por ejemplo, el tablero de la figura **??** tiene 6 superreinas y flechas con los movimientos que puede realizar la superreina en la posición (5,2):

   Se considera que una superreina está atacando a otra si es posible desplazarla hasta su posición. Por ejemplo, en el tablero anterior la $superreina_{(5,2)}$ está atacando a las superreinas en las posiciones (1,2), (3,2) y (1,6).

   Realiza un programa que permita determinar cuál es la superreina del tablero que está atacando la mayor cantidad de fichas. Para ello implementa la función `resolver(reinas)`, que recibe una lista de listas `[i,j]` con las posiciones de las superreinas y retorna la posición de la superreina que esté atacando más fichas. En caso de empate, retorne cualquiera de las superreinas involucradas en el empate.

   Observación 1: En el tablero de ejemplo la $superreina_{(5,2)}$ está atacando a la $superreina_{(1,6)}$ desde 4 direcciones distintas, pero cuenta como una sola ficha atacada.

   Observación 2: Su programa debe funcionar para un tablero de 8×8.

   Observación 3: Su solución no debería tener más de 50 líneas. Yo lo resolví en 34, pero la leyenda dice que Chuck Norris lo resolvió en 15 (sin usar `";"`)...

[1] Respuesta: 172
[2] Respuesta: 4.246
[3] Respuesta: 1.533.776.805

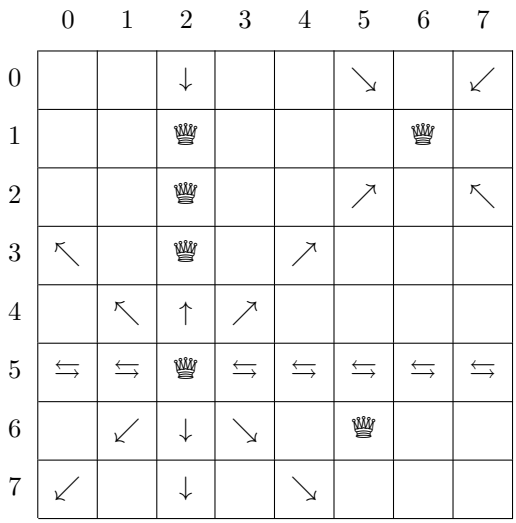

Figura 2: Tablero de ejemplo superreinas.

```
# Por ejemplo, para el tablero anterior el llamado a la función sería:
print(resolver([[1,2],[2,2],[0,5],[5,4],[6,3],[7,4],[5,2]]))
# ... y el retorno esperado es: [5,2]
```