Segunda Lista de Ejercicios

A continuación se presentan 6 ejercicios. Debe seleccionar al menos 4 para resolverlos y entregarlos. Cada ejercicio debe ir solucionado en un archivo diferente.

1. Se sabe que un polinomio puede ser representado por una lista. Por ejemplo, el polinomio $x^4 - 3x^3 - x + 4$ puede ser representado por la lista

$$[1, -3, 0, -1, 4\,]$$

Por lo general, algunas veces se representa este polinomio así:

```
1x^4 + -3x^3 + 0x^2 + -1x + 4
```

Escriba una función que se llame `ver_polinomio` que muestre de una mejor forma los polinomios, esto es, que no muestre el coeficiente 1, que no muestre un término con coeficiente 0 y que muestre correctamente los signos. Debe tener como argumento de entrada la lista con los coeficientes del polinomio y no debe retornar ningún valor, solo debe mostrar el polinomio.

Ejemplo de prueba: Si se ejecuta

```
P = [2, 1, 0, -1, -3, 2]

ver_polinomio(P)
```

Se debe mostrar

```
2x^5 + x^4 - x^2 - 3x + 2
```

2. Realice una función llamada $maxL$, que tiene como argumento de entrada un número entero positivo llamado n. Este número es el número de veces que el usuario puede ingresar un dato, los cuales deben ser almacenados. La función entonces calcula el mayor número de todos los datos ingresados y el número de veces que aparece, es decir, la función debe regresar estos dos valores.

Ejemplo de prueba: Si se ejecuta

```
A = maxL(4)

print("El mayor número es",A[0],"y se repite",A[1],"veces.")
```

Se debe mostrar

```
Ingrese un número: 8
Ingrese un número: 3
Ingrese un número: 8
Ingrese un número: 5
El mayor número es 8 y se repite 2 veces.
```

Nota: El usuario ingresó los datos 8, 3, 8 y 5

3. Suponga que *n* es un número natural y defina el siguiente proceso:

- Si *n* es par, entonces se calcula la mitad de *n*.
- Si *n* es impar, entonces se calcula *3n+1*.

Por lo tanto, dado un número cualquiera, podemos considerar su órbita al iterar el proceso varias veces. Por ejemplo, si *n=9*, al iterar varias veces el proceso tenemos:

- Como 9 es impar, entonces 3*9+1=28
- Como 28 es par, entonces 28/2=14
- Como 14 es par, entonces 14/2=7
- Como 7 es impar, entonces 3*7+1=22
- Como 22 es par, entonces 22/2=11
- Etc.

Entonces la órbita para *n=9* es 28, 14, 7, 22, 11, etc. La órbita de cualquier número *n* es una sucesión infinita.

Considere ahora *n = 8*:

- Como 8 es par, entonces 8/2=4
- Como 4 es par, entonces 4/2=2
- Como 2 es par, entonces 2/2=1
- Como 1 es impar, entonces 3*1+1=4
- Como 4 es par, entonces 4/2=2
- Etc.

En este caso la órbita es 4, 2, 1, 4, 2, 1, 4, 2, etc. Es decir, tenemos una órbita periódica.

La conjetura de Collatz dice que al realizar este proceso para cualquier número natural siempre alcanzaremos el número 1, y por tanto el ciclo 4, 2, 1.

Realice un programa que le pida al usuario ingresar un número natural cualquiera y retorne los elementos de su órbita en una lista hasta llegar al primer 1.

Ejemplo de prueba: Si el usuario ingresa 6, el programa debe mostrar

```
[3, 10, 5, 16, 8, 4, 2, 1]
```

4. Una forma de calcular la raíz cuadrada (positiva) de un número real a (positivo) es usando la ecuación

$$x_{n+1} = \frac{x_n + a}{x_n + 1}$$

Donde el valor inicial x_0 puede ser cualquier número real positivo. Por ejemplo, si $x_0 = 7$ y $a = 5$ entonces se obtiene la sucesión

$$x_1 = 1{,}5$$

$$x_2 = 2{,}6$$

$$x_3 = 2{,}111\cdots$$

$$x_4 = 2{,}285\cdots$$

$$x_5 = 2{,}217\cdots$$

$$x_6 = 2{,}243\cdots$$

$$x_7 = 2{,}233\cdots$$

$$x_8 = 2{,}237\cdots$$

$$x_9 = 2{,}235\cdots$$

$$x_{10} = 2{,}236\cdots$$

Nota:

$\sqrt{5} = 2{,}23606\ldots$

Haciendo uso de este procedimiento, realice un programa que primero le pida al usuario un número a al que se quiere calcular la raíz cuadrada, luego que pida el valor inicial x_0 y finalmente que pida la cantidad de cifras decimales c (entre 1 y 10) con la que se quiere calcular la raíz cuadrada. Además, el programa debe mostrar la cantidad de iteraciones realizadas para llegar al resultado.

Ejemplo de prueba: Si se ingresa los parámetros $a = 2$, $x_0 = 3$ y $c = 4$, se debe mostrar 1,4142 y además se debe indicar que se necesitaron 6 iteraciones para llegar a este resultado, pues note que en este caso se tiene:

$$x_1 = 1{,}25$$

$$x_2 = 1{,}4444\cdots$$

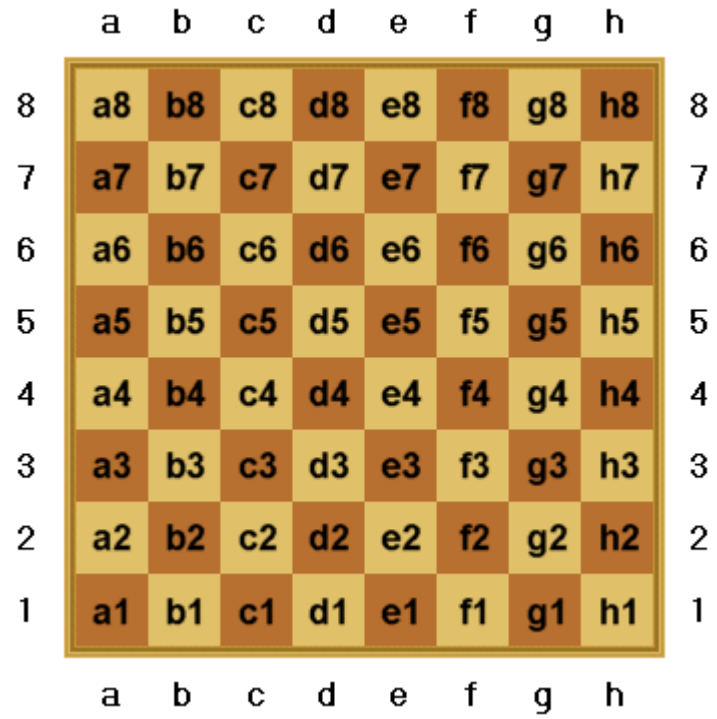
$$x_3 = 1{,}4090\cdots$$

$$x_4 = 1{,}4150\cdots$$

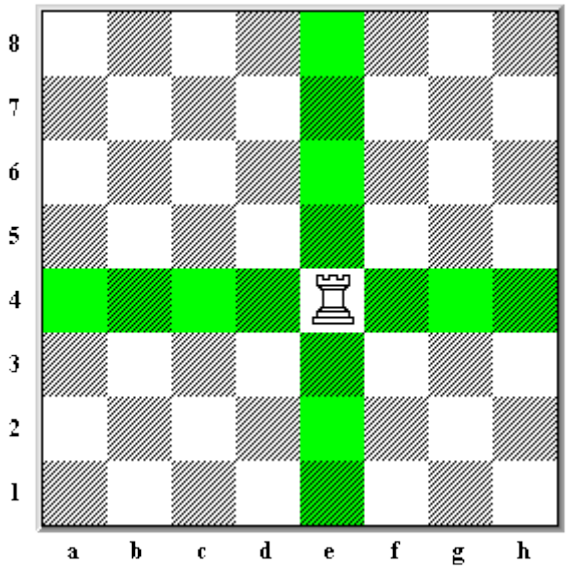
$$x_5 = 1{,}4140\cdots$$

$$x_6 = 1{,}4142\cdots$$

5. En el ajedrez, los escaques son las casillas cuadradas por donde se pueden mover las piezas. Cada escaque tiene una nomenclatura asociada, de la "a" hasta la "h" para indicar las columnas y desde el 1 hasta el 8 para indicar la fila.



Una torre es una pieza que solo puede moverse por la columna o fila en donde está.



Realice una función llamada `torre` que tenga como argumento de entrada la posición de la torre en el tablero en formato `string` y que retorne en una lista los escaques por los cuales puede moverse la torre (considere que no hay más piezas en el tablero).

Ejemplo de prueba: Si se ejecuta:

```
T = torre("e2")
print(T)
```

Se debe mostrar

```
['a4', 'b4', 'c4', 'd4', 'f4', 'g4', 'h4', 'e1', 'e2', 'e3',
'e5', 'e6', 'e7', 'e8']
```

6. En cierta empresa, con frecuencia se instalan y desinstalan computadores en diversas estaciones de trabajo. Cada computador tiene asociado un número serial, una dirección IP desde la cual se puede conectar a internet, un piso en el que se encuentra ubicado y un usuario. Se ha creado un diccionario para almacenar esta información:

```
Computadores = {
    "201A":["M0J345", "10.38.15.1", "segundo piso", "Juan"],
    "302A":["M0J478", "10.38.24.2", "tercer piso", "María"],
    "303A":["M0J251", "10.38.24.3", "tercer piso", "Felipe"]}
```

La llave del diccionario corresponde a una etiqueta que identifica la estación

de trabajo.

Realice una función que se llame `agregarPC` que agregue un nuevo computador en el diccionario. La función tiene tres argumentos de entrada: La etiqueta de la estación de trabajo, la lista con la información del computador nuevo y el diccionario que se quiere modificar. La función no retorna ningún valor y solo agrega el nuevo computador si la estación de trabajo no está en uso. Si la estación de trabajo ya está en uso, debe mostrar un mensaje que indique esto.

Ejemplo de prueba: Si se ejecutan las líneas

```
Computadores = {"201A":["M0J345", "10.38.15.1", "segundo piso", "Juan"], "302A":["M0J478", "10.38.24.2", "tercer piso", "María"], "303A":["M0J251", "10.38.24.3", "tercer piso", "Felipe"]}

DatosPCNuevo = ["M0J125", "10.38.15.4", "segundo piso", "Andrés"]

agregarPC("202A", DatosPCNuevo, Computadores)

print(Computadores)
```

Se debe mostrar

```
{'201A': ['M0J345', '10.38.15.1', 'segundo piso', 'Juan'], '302A': ['M0J478', '10.38.24.2', 'tercer piso', 'María'], '303A': ['M0J251', '10.38.24.3', 'tercer piso', 'Felipe'], '202A': ['M0J125', '10.38.15.4', 'segundo piso', 'Andrés']}
```