Fundamentos de Programación Para Ingeniería **Universidad de Santiago**
Guía nro. 6: Unidad 1 (PEP 1) Facultad de ingeniería
Prof: John Serrano
Correo: john.serrano@usach.cl
Ayudante: Vanina Correa
Correo: vanina.correa@usach.cl

Instrucciones:

- Para todos los problemas identifique los datos de entrada, los datos de salida y las operaciones necesarias sobre los datos de entrada para obtener los datos de salida.
- Para todos los programas utilice la siguiente estructura:

```
#CONSTANTES
instrucciones...
#ENTRADAS
instrucciones...
#PROCESAMIENTO
instrucciones...
#SALIDAS
instrucciones...
```

- Comente cada una de las líneas de los programas.
- Siga las buenas prácticas. Está prohibido utilizar Inteligencia Artificial para resolver u obtener una "ayuda" con los problemas (ChatGPT, Gemini, Bing, Github Copilot, entre otros).
- Para cada problema haga un archivo .py.
- El orden de los ejercicios no representa la dificultad. Usted puede resolver los ejercicios en el orden que prefiera.

Problema 1 (PEP 1 2016-2)

Una caja TATA es una secuencia de ADN encontrada en varios organismos vivos. Comienza siempre con "TATAAA". Siempre sigue con tres o más A's en múltiplos de tres, es decir, TATAAAAAA es una caja TATA, pues contiene 6 letras A, pero TATAAAAA no lo es, pues sólo contiene 5. Construya un programa en Python que reciba como entrada una secuencia de ADN como string y entregue la mayor secuencia de la caja TATA. Su solución debe funcionar para cualquier secuencia de ADN. Los datos del ejemplo son solo referenciales Por ejemplo:

- Para el string ATTTGATGATAATTTATAAAAAAAAAATGATAAATA
- La caja TATA está ubicada en: ATTTGATGATAATT**TATAAAAAAAAAA**ATGATAAATA
- y por lo tanto, el programa debería entregar: TATAAAAAAAAA

Considere que se debe entregar siempre la secuencia TATA de mayor tamaño en el texto.

Problema 2 (PEP 1 2017-1)

Un panagrama es una oración que contiene todas las letras del alfabeto, por ejemplo, en español, la oración:

"Whisky bueno: ¡excitad mi pequeña y frágil vejez!"

Es un panagrama, por otro lado, la oración:

"Por un momento estuve en un extraño sillón de felpa chirriante"

No es un panagrama, pues no están presentes las letras b, g, h, j, k, q, w, x, y, z.

Construya un programa en Python que reciba como entrada un string e informe al usuario si este es un panagrama o no. Considere para su implementación que:

- El texto recibido puede venir con mayúsculas y minúsculas
- El programa recibirá sólo oraciones en inglés, como por ejemplo *"The quick brown fox jumps over the lazy dog"*, por lo que ni los tildes, ni la letra ñ deben considerarse.

Problema 3 (PEP 1 2016-2)

La dirección meteorológica de Chile está buscando construir un programa para saber cuáles han sido las temperaturas más altas registradas durante el último año en la ciudad de Santiago. Para ello han construido una lista con 365 elementos, indicando las máximas registradas en grados Celsius, desde el 1 de enero de 2015 hasta el 31 de diciembre de 2015. La lista tiene el siguiente formato:

$$\text{temperaturas\_maximas} = [31.5, 35.3, 28.2, ..., 29.7]$$

Donde la posición 0 representa los 31.5 grados del 1 de enero, la posición 1 la máxima del 2 de enero y así sucesivamente hasta el último elemento que representa la máxima del 31 de diciembre. Para apoyar la labor de la dirección meteorológica de Chile, desea que construya un programa en Python la lista de temperaturas y que entregue las n (número ingresado por el usuario) temperaturas más altas del año. Es decir, si n = 15, el programa debería mostrar los 15 valores más altos registrados. Considere que su solución debe funcionar para listas de cualquier tamaño y valor, pues se desea aplicar esta solución a otras ciudades y otros períodos de tiempo.

Restricción: No se pueden utilizar ni funciones ni métodos nativos e importados propios de Python para ordenar, ni para obtener el máximos/mínimos valores de un objeto iterable.

Problema 4 (PEP nro. 1 2019-1)

Johnny es un lingüista estudiando cómo se utilizan las vocales en los distintos idiomas del mundo, en particular, según su teoría, idiomas basados en el latín, como el español, italiano y francés tienden a tener más palabras con vocales consecutivas, mientras que otras lenguas europeas tienen menos palabras con vocales consecutivas.

A fin de poder terminar su investigación Johnny necesita un método rápido de determinar la cantidad de vocales consecutivas dentro de una palabra, por lo que solicita a usted la construcción de un programa en Python que automatice el proceso, es decir, se recibe como entrada un string y se entrega como respuesta el largo del substring que contenga más vocales sin interrupciones, por ejemplo:

- Si la entrada fuese: 'groovy', el substring más largo de vocales es de largo 2, 'oo'
- Si la entrada fuese: 'oiseaux', el substring más largo de vocales es de largo 3, 'eau'
- Si la entrada fuese: 'hubschrauber', el substring más largo de vocales es de largo 2, 'au'

Considere que su entrada siempre estará compuesta solamente por caracteres alfabéticos sin tilde y en minúscula. Para su solución, no puede utilizar las palabras reservadas **for** e **in**.

Problema 5 (PEP nro. 1 2018-2)

Construya un programa en Python que, dados dos números enteros A y B y muestre como salida el mensaje 'Existen' si en el número A existen 3 copias seguidas del mismo dígito y también en B existen dos copias seguidas del mismo dígito. En caso de que esta condición no se cumpla, el programa debería mostrar el el mensaje 'No existen'. Por ejemplo:

- En este caso, el programa mostraría 'Existen', pues existen 3 9's en A y 2 9's en B
  - A = 4569993433212333
  - B = 6677991235
- En este caso, el programa mostraría 'No existen', pues si bien existen 3 2's en A, no existe un par del mismo dígito en B.
  - A = 45634322221233
  - B = 667512450003611
- En este caso, el programa mostraría 'No existen', pues no existen 3 dígitos repetidos continuamente en A.
  - A = 12345
  - B = 12345

Problema 6 (PEP nro. 1 2017-2)

Cuando escribimos algo erróneo en nuestros teléfonos, normalmente este tiende a corregirnos, sugiriéndonos la palabra que cree, se acerca más a nuestra palabra errónea. Una forma de determinar qué palabra es más cercana a la que hemos escrito, es la distancia Hamming. Esta métrica se obtiene calculando cuántos caracteres de diferencia tienen palabras de un mismo largo, por ejemplo:

- La distancia Hamming entre 101101001 y 101001111 es 3, pues hay 3 caracteres que difieren.
- La distancia Hamming entre tener y Tener es 1, pues hay 1 carácter que difiere.
- La distancia Hamming entre valorar y rarolav, es 4, pues hay 4 caracteres que difieren y sólo las letras o y a, están en la misma posición en ambos strings.

Utilizando la distancia Hamming, un teléfono compara la palabra que escribimos con las del diccionario, y obtiene todas las palabras con la menor distancia Hamming posible y luego, basándose en nuestro estilo de escritura, y la posición de las letras en el teclado, nos sugiere la palabra más cercana. A fin de implementar un sistema rudimentario de autocorrección, se le solicita a usted que construya un programa en Python que reciba como entrada dos strings del mismo largo y entregue como resultado la distancia de Hamming entre ambos.

Restricción: no se pueden usar las palabras reservadas **for** e **in** para el desarrollo del ejercicio.

Problema 7 (PA 2017-2)

Si consideramos dos conjuntos de enteros positivos $A = [a_0, a_1, \ldots, a_{n-1}]$ y $B = [b_0, b_1, \ldots, b_{n-1}]$. Se dice que un número entero positivo x existe 'entre los conjuntos' si:

- Todos los elementos de A son divisores exactos de x
- Todos los elementos de B son divisibles por x

Es decir, todo número divisible por cada elemento A y divisor de cada elemento de B, es un número que existe 'Entre Conjuntos'. Por ejemplo:

- Si $A = [4, 2]$ y $B = [16, 96, 32]$, los números que cumplen ambas condiciones son 4, 8 y 16, por lo tanto, los números que existen 'entre conjuntos' son 4, 8 y 16
- Si $A = [3, 5]$ y $B = [15, 30, 12]$, no existe un número que cumpla ambas condiciones, por lo tanto, no hay x que satisfaga dicha propiedad

Se requiere que construya un programa en Python que dados los conjuntos A y B, entregados como listas, indique el conjunto de números que cumplen con la propiedad de existir 'Entre Conjuntos'.

Problema 8 (PEP nro. 1 2019-1)

Un número fuerte es un número que cumple la condición de que la suma de los factoriales de sus dígitos son iguales al número original, por ejemplo:

- 2 es un número fuerte pues: $2! = 2$
- 145 es un número fuerte pues: $1! + 4! + 5! = 1 + 24 + 120 = 145$
- 23 no es un número fuerte pues: $2! + 3! = 2 + 6 = 8$

Construya un programa en Python que permita determinar si un número es fuerte o no. Considere que para su solución no puede utilizar funciones importadas (Contenido que no entra en la PEP 1).

Problema 9 (PA 2017-2)

Juanito es un lingüista que está investigando las redundancias en el lenguaje, para ello está intentando reducir las palabras a su mínima expresión, para ello ha consultado en sus referencias y un algoritmo que podría servirle es el de reducción de strings, el cuál es un algoritmo sencillo que funciona sólo con un par de reglas:

- Se pueden eliminar cualquier par de letras adyacentes, siempre y cuando estas sean iguales.
- Mientras existan dos letras iguales adyacentes, se deben seguir eliminando los pares hasta que no quede ningún par de letras iguales adyacentes.

Por ejemplo:

- Si la entrada fuese 'aabcc', el resultado sería 'b', pues se pueden eliminar las 'aa' del inicio y las 'cc' del final.
- Si la entrada fuese 'murcielago', el resultado sería 'murcielago', pues no hay letras iguales adyacentes para eliminar.
- Si la entrada fuese 'aaabccddd', el resultado sería 'abd' pues:
  - Se eliminan las 'aa' del inicio, quedando 'abccddd'
  - Se eliminan las 'cc' del medio, quedando 'abddd'
  - Se elimina un par de 'd's, quedando finalmente 'abd' que no puede reducirse más
- Si la entrada fuese 'aabccbaa', el resultado sería un string vacío pues.
  - Se eliminan las 'aa' del inicio, quedando 'bccbaa'
  - Se eliminan las 'cc' del medio, quedando 'bbaa'
  - Se eliminan las 'bb' del inicio, quedando 'aa'
  - Se elimina las 'aa' restantes, quedando un string vacío

Construya el programa en Python que implementa la reducción de strings para cualquier palabra ingresada. Considerando que:

- Si el resultado es un string vacío el programa debería informarlo.
- El programa debe funcionar para cualquier caso que respete el formato de entrada, no solamente para los ejemplos aquí dados

Problema 10 (PEP nro. 1 verano 2017)

La autora sueca Astrid Lindgren, autora de la famosa serie de libros acerca de Pippi Långstrump, escribió una serie de tres libros acerca de un niño detective, Kalle Blomkvist, y sus amigos. En estos libros, los niños resuelven misterios que encuentran o crímenes donde los adultos han pasado por alto detalles importantes. En esta serie, los niños utilizan una variante lúdica del lenguaje llamada Rövarspråket (lenguaje de ladrón), tanto para hablar en código como para jugar. Las reglas que utiliza el rövarspråket son sencillas: cada consonante se duplica y se inserta una 'o' entre las consonantes duplicadas.

Ejemplo:

Texto de entrada: 'esto es divertido'
Texto de salida: 'esostoto esos dodivoverortotidodo'

Este código no es muy útil en su forma escrita, pero un par de hablantes experimentados pueden convertir en indescifrable su conversación al utilizarlo, al estilo de la jerigonza en español. Se pide que construya un programa que traduzca un texto normal a rövarspråket. Considere que el texto es ingresado en minúsculas y las letras en él solo son las inglesas, es decir, ni tildes ni ñ.

Problema 11 (PEP nro. 1 2017-1)

La conjetura de Collatz es un problema aún no resuelto en matemáticas que postula que a partir de cualquier número entero positivo n es posible llegar al valor 1 con dos reglas sencillas:

- Si n es par, se divide por dos.
- Si n es impar, se multiplica por tres y se le suma 1.

Según esta conjetura, si se repite este proceso un número finito de veces, sin importar cuál sea el número inicial, se llegará al resultado final de 1. Actualmente la conjetura de Collatz está comprobada para todos los valores enteros menores a 5^{60}, por lo que se solicita a usted que construya un programa en Python que muestre el camino que se realiza para probar la conjetura con un entero n. Considere que la entrada es un número entero positivo.

Casos de ejemplo:

Entrada 1: 5
Salida 1:
5 16 8 4 2 1

Entrada 2: 7
Salida 2:
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Problema 12

En estadística uno de los gráficos más utilizados es el histograma, este es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados. Construya un programa en Python que recibiendo como entrada una lista de números enteros separados por espacios, dibuje un histograma utilizando un carácter.

Ejemplo:
Entrada: 2 4 3 5
Salida:

Problema 13

Las inversiones realizadas por una corporación financiera en diversos fondos durante el último año se mantienen registradas en una lista con el siguiente formato:

inversiones = [[Nombre del fondo (string), Monto de la inversión (int)], . . .]

Construya un programa en Python que muestre el monto total de las inversiones realizadas en cada fondo.

Ejemplo:
Entrada:

[['fondo A', 5000000], ['fondo B', 10000000], ['fondo A', 34000000], ['fondo A', 77000000], ['fondo C', 200000], ['fondo B', 21000000], ['fondo A', 36000000]]

Salida:
El monto invertido en fondo A es: 152000000
El monto invertido en fondo B es: 31000000
El monto invertido en fondo C es: 200000

N.B. utilizando la función nativa eval() es posible ingresar una lista por teclado, por ejemplo: eval(input()) convierte la secuencia de caracteres ingresada por teclado ['fondo A', 5000000] en un objeto tipo lista con dos elementos.

Problema 14 (PEP nro. 1 2019-1)

Un antiguo hechicero dejó en su tratado de magia '*Quæ est in vita magicis*' la teoría de que el poder de los hechizos reside en cómo se utilizan las vocales para construirlo. Según su teoría:

- Un hechizo es **pulentus** si todas las vocales que se utilizaron en su construcción aparecen en orden alfabético.
- Un hechizo es **bacanus** si las vocales que utiliza aparecen en orden alfabético inverso.
- Un hechizo es **cumast**, si las vocales que utiliza no tienen en orden alguno.

Por ejemplo:

- *occidis simia*, es **bacanus** pues las vocales que tiene están en orden alfabético invertido.
- *modicum cattus*, es **cumast**, pues las vocales no tienen orden alfabético alguno.
- *sedivid oud*, es **pulentus**, pues las vocales aparecen en orden alfabético

Construya un programa en Python que reciba un string y clasifique el hechizo según las condiciones que cumpla, considere que la entrada siempre contendrá sólo letras en minúscula. Considere que, si el hechizo tiene una sola letra, por ejemplo, *vini, vidi, vici*, cualquiera de las categorías sería correcta.

Problema 15 (PEP nro. 1 2019-1)

Ramujan es el líder de un temible ejército de soldados, su ejército jamás ha conocido una derrota en combate, sin embargo, ellos no son invencibles, pues cuando el fuego se interpone entre ellos y su enemigo, simplemente escapan de la batalla, sin luchar. Dadas estas condiciones, si tenemos la localización de los soldados de Ramujan, de los soldados enemigos y del fuego, podemos determinar cuántos soldados enemigos sobreviven. Para ello, supongamos que los soldados del ejército de Ramujan se identificarán con 'X', sus enemigos con 'O' y el fuego con '*'. Para determinar cuántos enemigos sobreviven consideraremos que, cada vez que exista un camino desde una 'X' hasta una 'O', sin fuego de por medio, el soldado de Ramujan elimina a su enemigo, sin importar si el soldado 'X' está superado numéricamente o no. Sin embargo, si existe fuego en el camino de 'X' hacia 'O', los soldados enemigos sobreviven al ataque. A fin de poder saber cuántos enemigos sobrevivirán al terrible ejército de Ramujan, construya un programa que reciba como entrada un string, representando el campo de batalla y entregue la cantidad de soldados enemigos que sobreviven.

Por ejemplo:

- 'X*OO*XX', resultaría en 2 sobrevivientes, pues los dos soldados enemigos están protegidos por el fuego.
- 'X*OX**XO*XOX', no tendría sobrevivientes, pues todos los soldados enemigos tienen al menos un soldado de Ramujan adyacente a él.
- 'X*OO*OO*X*OX*', tendría 4 sobrevivientes, pues existen 4 soldados protegidos por el fuego.

Considere para su solución que no es posible utilizar el método .count(), ni la palabra reservada in

Problema 16 (PEP nro. 1 2019-1)

Escritos antiguos en las paredes señalan que un número de propiedades mágicas puede ser encontrado utilizando la fórmula:

$$x = n^2 + (n+1)^2$$

Siendo n un entero positivo y obteniendo el 85^{vo} número primo que esta genera. Se sabe que esa fórmula puede generar números primos, pero también se sabe que no todos los números que genera lo son. Por ejemplo, para $n = 1$, se genera 5, que es un número primo, pero para $n = 3$ genera 25, que no lo es. Escriba un programa que encuentre el 85^{vo} número primo generado por esa fórmula, para descifrar el misterio.

Problema 17 (PEP nro. 1 2018-2)

Pablo es un programador aficionado que está buscando trabajo, Pablo sabe que para este tipo de trabajos, muchas veces es más importante el ingenio que la capacidad de escribir código, por lo tanto para asegurar ser uno de los mejores aspirantes al puesto, ha decidido crear un programa que le permita codificar palabras, para ello, ha inventado el siguiente algoritmo:

- Cambiar el primer carácter de la palabra, con el segundo, el tercero con el cuarto y así sucesivamente, si la palabra tiene un número impar de caracteres, el último carácter se mantiene igual.
- Reemplazar cada 'a', por una 'z', cada 'b' por una 'y', cada 'c' por una 'x' y así sucesivamente, hasta reemplazar cada 'z' por una 'a' en el mensaje obtenido en el primer paso.

Ayuda a Pablo, implementando su algoritmo en Python, considerando que:

- Las palabras siempre estarán escritas en inglés, es decir, no existen ñ's ni tildes y siempre las letras serán minúsculas.
- El programa aceptará como entrada una palabra y entregará como salida la palabra codificada.

Problema 18 (PEP nro. 1 2018-1)

Mr. Oso quiere comprar una pelota de fútbol y una camiseta y como el fanático del deporte que es, planea gastar el máximo posible para conseguir ambos ítems sin salirse de su presupuesto. A fin de conseguir su cometido Mr. Oso desea un programa en Python que reciba el presupuesto con el que actualmente cuenta, los precios de las pelotas de fútbol y los precios de las camisetas a fin de entregarle el valor total que le costará obtener la mejor combinación de ítems.

Por ejemplo, si Mr. Oso cuenta con un presupuesto de 60000 pesos y las pelotas de fútbol cuestan = [5000, 8000, 12000] y las camisetas cuestan = [40000, 50000, 60000], él podría comprar la camiseta de 40000 + la pelota de 12000 = $52000, o en cambio la camiseta de 50000 + la pelota de 8000 = $58000, en este caso la opción que más se acerca a su presupuesto es la segunda, por lo tanto el programa le informaría de esta opción.

Construya el programa que Mr. Oso necesita, para ello considere que se solicitan por teclado el presupuesto de Mr. Oso, la lista de precios de las pelotas de fútbol y la lista de precios de las camisetas. Considere que las entradas siempre contendrán números positivos, sin embargo, en caso de que ningún par pelota, camiseta esté dentro del presupuesto, el programa deberá indicar que se gastará $0. El programa debe funcionar para cualquier valor de presupuesto y precios de pelotas y camisetas, no solamente el ejemplo del enunciado.

Problema 19

Considere el registro de las calificaciones finales de los alumnos y las alumnas de un curso de programación en el siguiente formato: [[Nombre(String), Calificación(Float)]]

Construya un programa en Python que entregue las siguientes salidas:

- Lista con los nombres de los alumnos y las alumnas aprobados.
- Lista con los nombres de los alumnos y las alumnas que deben rendir examen ($3.0 \leq$ calificación ≤ 3.9).
- Lista con los nombre de los alumnos y las alumnas reprobados.
- Lista con nombre y nota de el o los alumnos con mejor calificación.
- Lista con nombre y nota de el o los alumnos con peor calificación.
- Calificación promedio del curso.

Ejemplo:
Entrada:

[['Felipe Echeverría', 2.2],['María Pérez', 5.6],['Rosa Navarro', 3.6],['Francisco Chávez', 5.0]]

Salida:
La lista de alumnos reprobados es: ['Felipe Echeverría']
La lista de alumnos que rinden examen es: ['Rosa Navarro']
La lista de alumnos aprobados es: ['María Pérez', 'Francisco Chávez']
La peor calificación es: [['Felipe Echeverría', 2.2]]
La mejor calificación es: [['María Pérez', 5.6]]
El promedio de las calificaciones es: 4.1

Problema 20 (Tarea nro. 1 verano 2017)

Aprovechando el caos de su nación, Robert Stroud ha decido escapar de Alcatraz. La prisión se encuentra en medio de la guerra producida entre las facciones del norte y el sur, por lo que Stroud considera que es la ocasión perfecta para ejecutar su plan. Él conoce la ubicación de las llaves de la prisión y de las puertas principales, así como las áreas de patrullaje de los guardias. El caos en la prisión es momentáneo, y no está del todo seguro de que tendrá tiempo suciente para ejecutar su plan.

Stroud es un maestro del sigilo y es capaz de recorrer un área de la prisión en 1 hora sin ser detectado. Si el área está custodiada por un guardia demora 2 horas en pasar sin ser detectado. Si Stroud necesita retroceder y pasar por un lugar que ya conocía, demorará la mitad del tiempo que necesitaba originalmente.

A partir de la información entregada, cree un programa en Python que permita a Stroud saber si puede ejecutar su plan o no. Para esto, considere que el programa recibirá como entrada la cantidad de horas que tiene disponible así como una lista que representa el mapa de la prisión. La lista contendrá caracteres, donde S representa a Stroud, C representa un camino libre, G representa un camino con guardia, L representa la llave buscada y E la salida de la prisión. Considere que Stroud siempre comienza en la primera posición de la lista, pero la llave, la salida y el o los guardias pueden estar en cualquier lugar. El programa debe indicarle a Stroud si con el tiempo ingresado logra escapar o no de la prisión. En caso de lograrlo, debe indicarse el tiempo utilizado y el tiempo que le sobra.

Por ejemplo, para la entrada ['S','C','G','E','L'] y 12 horas disponibles para el escape, se debe indicar que Stroud logra escapar en 5 horas y 30 minutos, restándole 6 horas y 30 minutos. Considere que esto es solo una entrada de ejemplo, su programa debe funcionar para cualquier lista que siga las instrucciones señaladas.

Problema 21 (Tarea nro. 1 verano 2017)

La asociación de Espías S.A. está creando un nuevo método de mensajes codificados. Su idea consiste en entregar un mensaje de varias palabras, donde solo un carácter de cada palabra es contenido en el mensaje final. Para conocer la posición del carácter a usar en el mensaje final, se utiliza el mensaje en si mismo, donde la cantidad de palabras indica la posición del carácter de cada palabra.

Con esta información, se solicita que cree un programa en Python que codifique mensajes utilizando el método creado por la asociación de Espías S.A. Para esto, su programa debe recibir un string que contendrá el mensaje original de n palabras (separadas por espacio) y se debe entregar el mensaje codificado, como string, para la salida.

Si la palabra que se codificará tiene menos caracteres que la posición buscada, se debe considerar hacer una búsqueda de manera circular y volver a recorrer la palabra desde la posición inicial, manteniendo la información de cuantas posiciones se ha recorrido hasta ese momento. Por ejemplo, si la palabra es hola y se debe avanzar a la posición 7 de la palabra, se debe considerar lo presente en la tabla nro. 1, con lo que la letra buscada será la *a*.

Palabra	*h*	*o*	*l*	*a*
Posición	0	1	2	3
	4	5	6	7

El siguiente es un ejemplo de codificación de un mensaje utilizando el método de Espías S.A.

Entrada:
'Este es el famoso curso de Fundamentos de Computación y Programación'

Salida:
'esloueFeCyn'

Problema 22 (PEP 1 2020-2)

Se le solicita construir un programa en Python en el cual el usuario pueda ingresar cuántas palabras desee y que entregue como resultado cuál es la palabra con la puntuación más alta, y el valor de esa puntuación. Para ello considere la siguiente tabla que muestra el valor asociado a cada letra, donde el símbolo "?" denota a un comodín (que puede reemplazar cualquier letra).

Letra	Valor	Letra	Valor	Letra	Valor
A	1	K	8	S	1
B	3	L	1	T	1
C	3	LL	8	U	1
CH	5	M	3	V	4
D	2	N	1	W	10
E	1	Ñ	8	X	8
F	4	O	1	Y	4
G	2	P	3	Z	10
H	4	Q	5	?	0
I	1	R	1		
J	8	RR	8		

Considere que el usuario sólo ingresará palabras con las letras aquí presentes. Si una palabra tiene alguna letra de dos caracteres ('CH', 'LL', 'RR') el puntaje debe calcularse solo como el valor asociado, sin sumar adicionalmente el valor de cada caracter por separado. En caso de que existan dos palabras que comparten el puntaje más alto, basta con mostrar una de las dos. Se prohíbe el uso de:

- Funciones y métodos para ordenar o para obtener máximos y mínimos.
- Ciclos for.
- Uso de strings con formato para imprimir, ya sea utilizando la sintaxis de llaves , o deporcentaje % (No se ve en el curso).
- Funciones propias, es decir, funciones definidas por el propio programador (No se ha visto aún).
- Programación orientada a objetos, es decir, definición de clases y métodos (No se ve en el curso).

Problema 23 (PEP 1 2021-2)

Camilo es un profesor de matemáticas que está enseñando a sus estudiantes el uso de fracciones y durante esta etapa ha notado que sus estudiantes han tenido problemas para revisar los resultados de los cálculos de los ejercicios que él propone. Cómo Camilo desea que sus estudiantes practiquen con ejercicios de fracciones hasta dominarlos a la perfección, para él sería de gran ayuda contar con una solución tecnológica que permita al estudiante escribir un ejercicio con fracciones y visualizar el paso a paso del desarrollo para comprobar si el resultado que hizo en papel es correcto o cometió errores en algún punto.

Para ayudarlo, se solicita a usted la construcción de un programa en Python que permita al usuario revisar el paso a paso de un ejercicio de fracciones a partir de un string que represente al ejercicio. Por ejemplo, si el estudiante quisiera ver el paso a paso del siguiente ejercicio:

$$\left(\frac{5}{6}+\frac{-4}{8}\right)*\left(\frac{9}{5}-\left(\frac{12}{5}*\frac{6}{4}\right)\right)$$

Ingresando la expresión:

```
(5/6 + -4/8) x (9/5 - (12/5 x 6/4))
```

debería poder ver el paso a paso de la resolución del ejercicio:

```
(1/3) x (9/5 - (12/5 x 6/4))
(1/3) x (9/5 - (18/5))
(1/3) x (-9/5)
-3/5
```

Para el desarrollo de su solución, considere que esta debe contemplar:

- Cálculos de expresiones que consideren operaciones de suma (+), resta (-), multiplicación (x) y división (:), representadas por los símbolos aquí indicados.
- Las expresiones complejas, es decir, que tienen más de un operador, deberán indicar SIEMPRE el orden de precedencia con paréntesis.
- Entre operadores y fracciones siempre existirá un caracter espacio, entre paréntesis y operadores también.
- La línea fraccionaria será siempre el caracter slash (/).
- Los numeradores siempre serán números enteros (positivos, negativos o cero) y denominadores siempre serán números enteros positivos.
- En caso de querer operar enteros con racionales, el entero debe escribirse de la forma entero/1, por ejemplo, el entero 9 debería escribirse como 9/1.
- Todo resultado, total o parcial, debe simplificarse antes de ser mostrado, es decir, valores como 3/6 deberían visualizarse como 1/2.
- Cómo los números flotantes (float) tienen imprecisiones, no deberían usarse para el cálculo de los resultados.