1.

Introducción

Ahora que sabes que eres un/a semidiós/a, decides hacer lo posible porque tu nombre suene en el Olimpo. Un día, mientras te diriges a Heraclio, al principal puerto de DCCreta, te encuentras con un individuo de apariencia extraña que está muy desorientado. Te acercas a preguntarle quién es y si necesita ayuda, momento en que te percatas de algo muy curioso de su vestimenta: sus sandalias tienen alas. Ya has escuchado de estos míticos objetos antes: no hay duda alguna de que este individuo es Hermes, hijo de Zeus. Hermes es el dios de los mensajeros, de las fronteras y los viajeros que las cruzan, del ingenio y del comercio en general, de la astucia, de los ladrones y los mentirosos, y el que guía las almas al inframundo.

Resulta que unos cíclopes molestosos des-calibraron el báculo (bastón) de Hermes, el que le indicaba la distancia y dirección hacia donde tiene que llevar sus importantes encomiendas. Esta es la oportunidad perfecta para darte a conocer con los dioses del Olimpo: si ayudas a Hermes entonces se sorprenderá gratamente y entregará el mensaje de tu existencia a los demás dioses.
Entonces, te ofreces a ayudarle. Hermes, te explica cómo es que él determina donde van sus pedidos:
> Sin mi báculo, no tengo la dirección que debo tomar, ni cuan lejos debo ir. Mis encomiendas tienen un único mensaje encriptado que puedo traducir con mi báculo, sin embargo, no estoy seguro de como descifrarlo.
Entonces, después de analizar varios paquetes, encuentras un patrón. Ahora solamente queda crear una herramienta que permita descifrar. Afortunadamente, se cruza en el camino una Pitón (en inglés, Python), y recuerdas como desde una temprana edad lograbas hacer cosas inimaginables con estos reptiles a través de un lenguaje extraño (Python).

Objetivo

En esta primera parte de la tarea, deberás crear 2 funciones que reciban 1 `string` como parámetro y retornen lo pedido; y también recibirás una cantidad **N** (número indicado por input) de frases a las cuales les deberás implementar las 2 funciones para ayudar a Hermes.
Las funciones por crear son las siguientes:

- **distancia(string):** Esta función tiene como objetivo **calcular la distancia que debe indicar el báculo** de Hermes. Recibe un parámetro `string`, el cual corresponderá a una palabra o frase. Deberás contar la cantidad de vocales de cada tipo que hayan (cantidad de letras **A**, cantidad de letras **E** y lo mismo con **I, O, U**). Debes **detectar tanto mayúsculas como minúsculas** (no habrán tildes, no te preocupes de ello).
  Una vez hecho esto, deberás retornar un número *int* a según las siguientes reglas **(deben estar priorizadas en este orden)**, el cual corresponde a la distancia a recorrer:

Casos	**Retorno (int)**
Si hay exactamente 2 letras **O** y más de 1 letra **E.**	El número 13
Si hay más de 7 letras **U**	La cantidad de letras u
Si hay 1 vocal o más de cada tipo (**A**, **E**, **I**, **O**, **U**)	La parte entera de la raíz cuadrada de la suma entre las cantidades de letras **A**, **E**, **I**, **O** y **U.**
En cualquier otro caso	La cantidad de letras **A** más las letras **E** más 1 y todo eso al cuadrado.

- **direccion(string):** Esta función tiene como objetivo **determinar la dirección en que se debe mover Hermes**. Recibe un parámetro `string` que corresponde a una palabra o frase, para retornar la dirección a seguir. Esta última se determina según la **primera letra especial (N, S, E o la letra O)** que se encuentre en el parámetro recibido, de izquierda a derecha. Si se encuentra la letra N, S, E o la letra O, se debe retornar un `string` igual a `"norte"`, `"sur"`, `"este"` o la palabra `"oeste"`, respectivamente. En caso de **no encontrarse ninguna de dichas letras**, deberás retornar el `string` que diga `"al inframundo"`. Debes **detectar tanto mayúsculas como minúsculas** (no habrán tildes, no te preocupes de ello).

Luego deberás implementar las funciones para ayudar a Hermes, recibirás N mensajes y para cada uno de ellos deberás sacarle la distancia y dirección encriptada en el mensaje (`string`). A continuación deberás imprimir como se explica abajo.

Input format

Recibirás primero un número entero que te dirá la cantidad de frases a recibir a continuación y luego recibirás esa cantidad de frases. A cada una de esas frases deberás sacarle la distancia y dirección oculta en ella.

Constraints

- No recibirás `strings` con tildes.
- **Deberás considerar las letras sin importar si son mayúsculas o minúsculas.**
- Para cada `string` debes calcular la distancia y dirección. Correspondiente.
- En el caso de la función distancia siempre habrá que priorizar según el orden en el que aparecen en la tabla explicativa. Es decir, si cumple con la primera y tercera condición, se le da prioridad a la primera (por estar antes en la tabla).

Output format

Una vez que calculas la distancia y direcion de un `string`, deberás imprimir la frase:

```
 "Hermes debe moverse {valor_distancia} metros en direccion
{valor_direccion}"
```

Siendo `valor_distancia` y `valor_direccion` los cálculos hechos para la distancia y dirección de cada string.

Ejemplo

Público #1:

- **Input:**

```
2
No te preguntare como estas (>_<) !!
Umpa lumpa, umpa lumpa, umpa lumpa, umpa lumpa, umpa lumpa, umpa lumpa
```

- **Output:**

```
Hermes debe moverse 49 metros en direccion norte
Hermes debe moverse 12 metros en direccion al inframundo
```

Como el entero que recibimos es 2, sabemos que recibiremos 2 frases.
Luego se calcula la distancia y dirección de la primera frase. Como no cumple ninguna de las condiciones de los casos de distancia se va a la última opción donde como hay 2 **A** y 4 **E,** se calcula (2 + 4 + 1) ** 2 = 49. Como la primera letra es **N,** `"norte"` será la dirección.
En la última frase (para entender su significado recomiendo ver este video), hay 12 letras **U** (entre minúsculas y mayúsculas) por ello cumple con el segundo caso y su distancia son 12 metros. De ahí como no tiene ninguna de las letras **N, S, E, O** su dirección será `"al inframundo"`.

2.

Introducción

Luego de ganarte el favor de Hermes, finalmente llegas a Heraclio. Una vez allí, se te aparece una mujer con vestimenta de guerra, quien se presenta como Atenea. Mientras te dice eso, piensas para dentro: "*Atenea... ¡La Diosa de la sabiduría y estrategia en combate! Si me gano su favor de seguro me querrán en el Olimpo!*" Por lo que te dispones a ayudar como sea.
Ella te informa que Hermes ha logrado volver al monte Olimpo con una pitón en la mano, y le contó a todos los dioses presentes el cómo tú los habías ayudado. Ahora, ella había sido enviada para probar y comprender estos conocimientos en programación de pitones. Es por esto que se comunica contigo y solicita tus habilidades para usar el desconocido poder de estos reptiles en sus propias misiones. En concreto, quiere evaluar la posibilidad de anticipar y predecir el poder de posibles contrincantes en distintos escenarios bajo incertidumbre.

Objetivo

En esta parte deberás crear 2 funciones y luego el flujo principal del programa (Recibir input, llamar a una de las funciones que crearás e imprimir un output determinado). Dentro de las funciones que deberás crear se encuentran, en primer lugar, la función **calcular_poder** que calcula el poder de un enemigo en un lugar específico. En segundo lugar la función **contar_letras_repetidas** que cuenta la cantidad de letras repetidas presentes en una palabra.
Contarás con una librería llamada **olimpo.py**, que tiene la función **clasificación**, que funciona de la siguiente manera:

- **clasificacion(nombre):** Esta función clasifica el nombre de un enemigo dentro de 4 categorías. Recibe como parámetro `nombre` [*str*] y retorna un *str* que corresponde a la categoría en la que se encuentra. La función solo retorna categorías presentes en el siguiente listado.

Categoría
"Olimpo"
"Olimpo fruna"
"Semi Dios"
"Mortal"

Por ejemplo si se llama a **clasificacion**, pasándole como parámetro "Zeus" como este dios está dentro de los 12 grandes dioses del olimpo la función retornaría el siguiente string.

```
"Olimpo"
```

Las funciones que debes implementar son las siguientes:

- **contar_letras_repetidas(lugar)**: Esta función cuenta la cantidad de letras repetidas presentes en una palabra. Recibe como parámetro `lugar` [*str*] y retorna un *int* que representa la cantidad de **letras** repetidas del string (El string puede contener elementos que no sean letras). Si en un string existen 2 letras iguales, entonces esa letra se repite 2 veces. Además, si en un string aparece una letra en mayúscula, y la misma letra en minúscula, entonces esa letra también se repite 2 veces. Por último si `lugar` no posee letras repetidas tiene que retornar 1.

  Por ejemplo si se llama a la función con el parametro "Peloponeso", se repite 2 veces la p, 2 veces la e y 3 veces la o por lo que la cantidad de letras repetidas totales sería 2 + 2 + 3 = 7. Las letras "l", "n" y "s" no suman nada porque no se repiten. La función retornaría el siguiente número.

  ```
  7
  ```

- **calcular_poder(nombre, lugar)**: Esta función calcula el poder de un enemigo en base a su nombre y el lugar del enfrentamiento. Recibe el parámetro `nombre` [*str*] y `lugar`[*str*], y retorna un *float* que representa el poder de un enemigo. La fórmula para calcular el poder es la siguiente:

  $$((habilidad\_de\_combate\_enemigo + aleatoriedad) \cdot buff)/c$$

  La **habilidad_de_combate_enemigo** se determina por categorías.

Categoría	**habilidad_de_combate_enemigo**
"Olimpo"	5
"Olimpo fruna"	3
"Semi Dios"	2
"Mortal"	0

Deberás importar y ocupar la función **clasificacion** para obtener la categoría en la que se encuentra el oponente
La **aleatoriedad** es un número entero al azar entre 0 y 2, ambos inclusive.
El **buff** es por defecto 1, sin embargo hay 4 grandes dioses del olimpo que poseen **buff** distinto. Si coincide el nombre del enemigo y el lugar con una fila de la tabla se concede el buff indicado (a excepción de "Ares" que **SIEMPRE** tiene buff).

Nombre	Lugar	Buff
"Zeus"	"Olimpo"	10
"Poseidon"	"Mar"	8
"Hades"	"Infierno"	7
"Ares"	**SIEMPRE TIENE BUFF**	15

Finalmente **conocimiento_lugar** se determina con la cantidad de letras repetidas que posee el lugar del enfrentamiento, lo que se calcula con la función **contar_letras_repetidas.** Si el lugar no posee letras repetidas el conocimiento del lugar es 1.

Input format

Recibirás una sola línea de input que tendrá el siguiente formato:

```
{nombre};{lugar}
```

Constraints

- Para poder usar la función **clasificación** debes importarla
- En la función **contar_letras_repetidas**, si el lugar no posee letras repetidas el conocimiento del lugar es 1.
- En la función **contar_letras_repetidas**, si en el string que se pasa como parámetro aparece una letra mayúscula y la misma letra en minúscula, entonces esa letra se repite 2 veces.

Output format

Deberás imprimir una sola línea con el siguiente formato:

```
El poder de {nombre} en {lugar} es {poder}
```

¡Recuerda tener cuidado con las tildes, mayúsculas y minúsculas!, ya que si el output tiene una mínima diferencia con el esperado será considerado incorrecto.

Importante

Debes ocupar exactamente los **mismos nombres de las funciones** que se explican en Objetivo. Además, las funciones **no deben recibir ni más ni menos parámetros** que los indicados y **deben recibirlos en el mismo orden**.
En este caso, las funciones son las siguientes:

- **calcular_poder(nombre, lugar)**
- **contar_letras_repetidas(lugar)**
- **clasificacion(nombre)**

Si no se siguen estas indicaciones, **tu código no podrá ser revisado.**

Ejemplo

Público #1:

En este ejemplo se recibe de input "Zeus" y "Olimpo", por lo que al calcular la **habilidad_de_combate_enemigo** se asigna un valor = 5 porque "Zeus" pertenece a la categoría de "Olimpo" y como se encuentra en su territorio recibe un buff = 10. Como "Olimpo" posee 2 letras repetidas **conocimiento_lugar** = 2. Asumiendo para este ejemplo que el valor aleatorio es 2 la fórmula de **calcular_poder** nos quedaría:
((5+2)*10)/2=35.0

- **Input:**

```
Zeus;Olimpo
```

- **Output:**

```
El poder de Zeus en Olimpo es 35.0
```

Público #2:
En este ejemplo se recibe de input "Teseo" y "Grecia", por lo que a la **habilidad_de_combate_enemigo** se le asigna un valor = 2 porque "Teseo" pertenece a la categoría de "Semi Dios", recibe un buff de 1 porque no es especial. Como "Grecia" no posee letras repetidas **conocimiento_lugar** = 1. Como en este ejemplo la aleatoriedad es 1, la fórmula de **calcular_poder** nos quedaría:
((2+1)*1)/1=3.0

- **Input:**

```
Teseo;Grecia
```

- **Output:**

```
El poder de Teseo en Grecia es 3
```

3.

Introducción

Una vez ganado el favor de Atenea, te dispones a cruzar desde DCCreta hacia el Hellas Continental, rumbo al monte Olimpo, por lo que debes comenzar navegando por el mar de DCCreta. Luego de unos días de viaje, un temporal derriba tu barco y caes al mar. Para tu sorpresa, y después de pensar que lo habías perdido todo, abres los ojos bajo el agua, y ¡recuerdas que puedes respirar!
Mientras estás intentando descifrar nuevamente este suceso, se aparece una figura humana nadando hacia ti:
> ¿Estoy Muerto? - le preguntas.
> Imposible, no dejaría derramar sangre divina en mis aguas.
> ¿Acaso eres tú, Poseidón?
> Así es. Si quieres llegar al Monte Olimpo, debes escuchar atentamente tu desafío en el mar de DCCreta.
Te informa que para llegar a Atenas, debes poder comunicarte con las criaturas marinas para obtener las direcciones hacia donde ir. Distintas criaturas tienen distintos lenguajes y crees que no podrás comunicarte debajo el agua, sin embargo, descubres que las puedes entender a *casi* todas. La única dificultad es con los cetáceos. "Estas criaturas son descendientes de Ceto, un temido monstruo acuático femenino" agrega Poseidón mientras te explica tu obligación de entenderlas, si quieres cruzar a la otra orilla con vida.

Objetivo

En esta tercera parte de la tarea deberás crear una función para comunicarte con un tipo de criatura marina: **decodificar_cetaceo**. Esta función se encarga de decodificar una frase en cetáceo a una en lenguaje legible.
La función a crear es la siguiente:

- **decodificar_cetaceo(frase_cetacea)**: Por medio de esta función se podrá **traducir una frase en cetáceo** a una legible por nosotros los humanos. Esta función recibe el

parámetro `frase_cetacea` y retorna un *string* correspondiente a la traducción de la frase cetácea.

La decodificación consiste en:

- Simplificar las palabras de la frase a palabras que ocupen solo un caracter por **letra o signo**, es decir, que **no contengan carácteres repetidos contiguos**. En el caso de ser letras, estas **se consideran iguales sin importar si están en mayúsculas o minúsculas**. Por ejemplo, la letra equis es igual si está escrita en minúscula "x" o en mayúscula "X".
- En el caso de que `frase_cetacea` tenga **números**, tanto uno como un grupo de ellos contiguos, deberá(n) ser **reemplazado(s) por solo un espacio**.
- Entregar la frase decodificada final en **minúsculas**.

La función termina **retornando la frase cetácea decodificada.**
Por ejemplo, podrías recibir el siguiente string como `frase_cetacea`:

```
¡aaAAA3389mmmMMiImmMMMmirRRRrR!!!!
```

Y en este caso, la frase retornada sería:

```
¡a mimir!
```

Input format

Tu misión únicamente es crear las funciones. Los inputs de los testcases son simplemente para que puedas visualizar cómo se está evaluando el testcase. **NO DEBES RECIBIR INPUTS.**

Constraints

- Para la función **decodificar_cetaceo,** la frase retornada, es decir, la frase decodificada final, **nunca tendrá carácteres iguales contiguos.** Por ejemplo, no se esperará que decodifiques una frase y esta contenga la palabra "llave", ya que esta tiene una doble ele.
- El *string* `frase_cetacea` nunca comenzará o terminará con un número. Tampoco contendrá espacios.

Output format

Tu misión únicamente es crear las funciones, ya que los outputs se generarán por detrás en el main. **NO DEBES IMPRIMIR NADA.**
¡Recuerda tener cuidado con las tildes, mayúsculas y minúsculas!, ya que si el output tiene una mínima diferencia con el esperado será considerado incorrecto.

Importante

Debes ocupar exactamente el **mismo nombre de la función** que se explica en Objetivo. Además, la función **no debe recibir ni más ni menos parámetros** que los indicados y **debe recibirlos en el mismo orden**.

En este caso, la función es la siguiente:

- **decodificar_cetaceo(frase_cetacea)**

Si no se siguen estas indicaciones, **tu código no podrá ser revisado.**

Ejemplo

Ejemplo #1:

- **Input:**

```
ggggrRRRRieEegggGGGooOooO6>5oOooiikoOOoOos
```

- **Output:**

```
griego > oikos
```

La primera línea en la sección **Input** corresponde al parámetro `frase_cetacea` de la función. Inmediatamente después, se puede ver en la primera línea de **Output** que el resultado de la decodificación es la frase "griego > oikos".
Esta frase no tiene ningún carácter contiguo repetido, como se ha explicado anteriormente, y su construcción se basa en dejar solo una letra minúscula o signo en lugar de muchos carácteres juntos. Además, las palabras se encuentran separadas por un espacio, en vez de por número(s) como en la frase original en **Input**.
En la siguiente tabla se desglosan las correspondencias de los carácteres en la frase decodificada final con respecto a la frase original:

Carácter final	Substring en la frase original
g	gggg
r	rRRRR
i	i
e	eEe
g	gggGGG
o	ooOooO
	6
<	<
	5
o	oOoo
i	ii
k	k
o	oOOoOo
s	s

4.

Introducción

¡Enhorabuena! Has logrado dominar el antiguo idioma de los cetáceos y finalmente tocas tierras continentales, luego de unos cuantos días en el mar. Agotado, comienzas a caminar hacia la capital de Grecia, Atenas, una de las paradas importantes antes de llegar al Monte Olimpo. Comienzas a divisar las primeras construcciones y te percatas de que hay focos de incendios en distintos puntos de la ciudad.
"No puede ser... Es Ares" piensas con todas tus esperanzas de descansar destruidas. Te enteras que Ares, el dios de la guerra, desató la ira contra su hermana Atenea atacando la ciudad. Los rumores dicen que una pitón, con la ayuda de un tal semidios, ayudaron a la diosa a estrategizar la guerra, lo cual es totalmente contrario a la violencia salvaje de Ares. Sin otra opción, te enlistas en las tropas de los aldeanos de Atenas, con el claro objetivo de sortear este obstáculo y seguir con tu camino.

Objetivo

En este punto de la tarea deberás simular la pelea de Ares con los aldeanos de la histórica ciudad de Atenas. Para ello, deberás crear dos funciones: **ataque_ares** y **calcular_estado_atenas**.
Las funciones a crear son las siguientes:

- **ataque_ares(lista_tropa, odio_del_turno, arma_ares)**: Esta función simula un ataque de Ares a la tropa de Atenas. Recibe los parámetros `lista_tropa` tipo *lista*, `odio_del_turno` tipo *float* y `arma_ares` tipo *str ing*, para retornar una *lista* con el estado de los aldeanos luego del ataque de Ares.

  El primero de los parámetros corresponde a una **lista de *strings*, cada uno de ellos con información de un aldeano** perteneciente a la tropa. Cada *string* se verá de la siguiente manera:

```
nombre_X,puntos_de_vida_X,fuerza_vital_X
```

Siendo nombre_X, puntos_de_vida_X y fuerza_vital_X el nombre, los puntos de vida y la fuerza vital del aldeano X respectivamente.
En cada ataque (es decir, cada vez que se llama a la función) Ares posee un puntaje, el cual se calcula entregándole el arma de Ares a la función importada **calcular_puntaje_ares** (ver Funciones entregadas).
Ares ataca a los primeros n aldeanos de la tropa, donde n se obtiene multiplicando `odio_del_turno` por la cantidad total de aldeanos de la tropa (usar **round**).
En el caso de que **n sea igual a cero**, es decir, Ares no intente atacar a nadie, se imprimirá:

```
Ha reinado la paz en Atenas (por ahora)
```

Por cada aldeano atacado, se debe calcular su puntaje, entregándole su nombre a la función importada **calcular_puntaje_aldeano** (ver Funciones entregadas). Además, se debe comparar su puntaje con el de Ares:
Si el puntaje de Ares es mayor que el del aldeano, Ares ataca al aldeano y éste último pierde vida igual a su fuerza vital. Debes imprimir:

```
Ares ha atacado al aldeano X y ha quedado con Y puntos de vida
```

Siendo X el nombre del aldeano atacado e Y la vida del aldeano después de ser atacado.
Si el puntaje de Ares es menor o igual que el del aldeano, éste último recupera vida igual a su fuerza vital. En este caso, debes imprimir:

```
El aldeano X ha probado la inmortalidad y ahora tiene Y puntos de vida
```

Siendo X el nombre del aldeano beneficiado e Y la vida final del aldeano.
En el caso de que el aldeano termine **sin puntos de vida**, dicho aldeano sale de la tropa final y deberás imprimir:

```
Se ha reportado el descenso del aldeano X
```

Siendo X el nombre del aldeano atacado.
Finalmente, debes **retornar la lista de la tropa actualizada**. Esta lista debe **respetar el orden de aldeanos de la lista original y reflejar los cambios en puntos de vida de los aldeanos correspondientes**. Solo en el caso del descenso de alguno de ellos, se deberá disminuir la cantidad de elementos en ella.

Para comprender mejor, a continuación se presentará un **ejemplo** de un llamado a la función con los siguientes parámetros:

- `['agatha,106,14', 'briseida,92,26', 'aquiles,102,11']`
- `0.6`
- `pistola`

La cantidad de aldeanos atacada en esta ocasión es 2, ya que la cantidad de aldeanos de la tropa es 3 y el odio del turno es 0.6, que al multiplicarse y redondearse a la unidad resulta 2. Se utiliza la función importada **calcular_puntaje_ares** para calcular el puntaje de Ares y este obtiene 4 puntos por atacar con el arma `pistola`. Se calcula el puntaje para cada uno de los aldeanos afectados, entregando su nombre a la función **calcular_puntaje_aldeano.**
Ares ataca al primer aldeano, ya que el puntaje de Ares es mayor que el obtenido por el aldeano `agatha`, el cual es 3. De esta manera, el aldeano atacado pierde 14 puntos de vida (su fuerza vital):

```
Ares ha atacado al aldeano agatha y ha quedado con 92 puntos de vida
```

Luego, el segundo aldeano obtiene igual puntaje que Ares, es decir 4, por lo que gana 26 puntos de vida. Como sus puntos de vida eran 92, termina con 118 puntos:

```
El aldeano briseida ha probado la inmortalidad y ahora tiene 118 puntos de vida
```

Finalmente, la lista que retorna este ejemplo es la siguiente:

- `['agatha,92,14', 'briseida,118,26', 'aquiles,102,11']`

- **calcular_estado_atenas(lista_tropa, estado_actual_atenas)**: Esta función calcula el estado de Atenas. Recibe los parámetros `lista_tropa` tipo *lista* y `estado_actual_atenas` tipo *int*, para retornar un *int* representativo del nuevo estado de la ciudad.

  Debes calcular el promedio de puntos de vida de la tropa (usar **round**) para obtener el nuevo estado de Atenas:

  - Si el promedio es mayor que 100, el nuevo estado de Atenas es `estado_actual_atenas` más el promedio de la tropa.
  - Si el promedio es menor o igual que 100, el nuevo estado de Atenas es `estado_actual_atenas` menos el promedio de la tropa. En el caso de que ésta resta sea menor que cero, se retorna 0 (el estado nunca puede ser negativo).
  - Si el promedio es cero, que es lo mismo que la tropa no tenga aldeanos, el nuevo estado de Atenas es 0.

  Para la misma tropa presentada en el ejemplo de la explicación de la función **ataque_ares,** se calculará el estado de Atenas posterior al ataque mostrado. Los parámetros recibidos por la función **calcular_estado_atenas** son los siguientes:

  - `['agatha,92,14', 'briseida,118,26', 'aquiles,102,11']`
  - `300`

  El cálculo del promedio de la tropa resulta igual a 104, ya que la suma de los puntos de vida de la tropa es 312 y el largo de la tropa es 3, que al dividirse y redondearse a la unidad resulta 104. Como 104 es mayor que 100, el estado final de Atenas corresponde a la suma del promedio de la tropa más el estado de Atenas recibido como parámetro, es decir, 104 más 300:

```
El estado actual de Atenas es 404
```

****

Importante: Los números de tipo *float*, específicamente los **n aldeanos atacados** y el **promedio de vida de la tropa**, deberán estar **aproximados a la unidad**. Para esto puedes

utilizar la función **round(float)**, que aproxima el valor de `float` a la unidad. Por ejemplo: `round(1.5)` retornaría `2`.
****
****

Input format

Tu misión únicamente es crear las funciones. **NO DEBES RECIBIR INPUTS.**

Constraints

- El valor `estado_actual_atenas` es un número entero no negativo. Su **mínimo valor es cero.**
- Ares siempre intentará atacar a los **primeros n aldeanos en `lista_tropa`.**
- Se debe **respetar el orden de la lista de la tropa inicial en todo momento**. La función **ataque_ares** debe recibir a `lista_tropa` y retornar una *lista* con los aldeanos en el mismo orden, aunque esta haya sufrido modificaciones y/o sustracciones de elementos.
- Tanto los nombres de los aldeanos como las armas de ares siempre serán *strings* con todas sus letras en minúsculas.
- Para la función **calcular_estado_atenas**, si la **lista de aldeanos está vacía** al momento de calcular el promedio de la tropa, este debe ser igual a cero.

Output format

Se te entregará un fragmento de código, el cual hará el llamado de las funciones y simulará la guerra entre la tropa de Atenas y Ares. Éste se llamará **flujo del juego y NO DEBES PROGRAMARLO**. Este consiste de turnos y en cada uno de ellos se llamará una vez a la función **ataque_ares** y luego una vez a la función **calcular_estado_atenas**, hasta que Ares o la tropa de aldeanos gane.
Todos los outputs explicados en esta sección serán los correspondientes al flujo del juego. El resto de outputs que verás en los testcases son resultado de los print dentro de las funciones que tú debes programar y están explicados en Objetivo.
Al comienzo del turno se imprimirá la línea:

```
Antes de la llegada de Ares, Atenas tenia el estado X
```

Siendo X el estado original de Atenas antes de comenzar la pelea.
Luego, al comienzo de cada turno, se imprimirá la línea:

```
El odio de este turno es X y Ares atacara con un(a) Y
```

Siendo X el odio del turno e Y la arma de Ares del turno.
Al terminar cada turno, se informará el estado actual de la ciudad después de ser atacada en la siguiente línea:

```
El estado actual de Atenas es X
```

Siendo X el estado de Atenas después del turno correspondiente.
Los aldeanos de Atenas pelearán mientras el estado de la ciudad sea menor o igual al estado original de la misma y mayor a cero.
En caso de que **`estado_actual_atenas` sea igual a cero, Ares ganará la pelea** y se imprimirá:

```
Los aldeanos no pudieron hacer nada contra la furia de Ares y Atenas quedo destruida
```

Por el contrario, la **tropa de Atenas ganará si `estado_actual_atenas` logra ser mejor que el estado original de la ciudad**. En dicho caso, se imprimirá:

```
¡Han ganado las tropas de Atenas!
```

¡Recuerda tener cuidado con las tildes, mayúsculas y minúsculas!, ya que si el output tiene una mínima diferencia con el esperado será considerado incorrecto.

Funciones entregadas

Se les entrega el módulo **ares**, el cual tiene las siguientes funciones que deben usar en esta parte:

- **calcular_puntaje_ares(arma_ares)**: esta función recibe como parámetro un arma, que es una palabra de tipo *string,* y retorna un *int* que representa el puntaje que le corresponde a la misma.

- **calcular_puntaje_aldeano(nombre_aldeano):** esta función recibe como parámetro un nombre de aldeano, que es una palabra de tipo *string,* y retorna un *int* que representa el puntaje que le corresponde al mismo.

Importante

Debes ocupar exactamente los **mismos nombres de las funciones** que se explican en Objetivo. Además, las funciones **no deben recibir ni más ni menos parámetros** que los indicados y **deben recibirlos en el mismo orden**.
En este caso, las funciones son las siguientes:

- **ataque_ares(lista_tropa, odio_del_turno, arma_ares)**
- **calcular_estado_atenas(lista_tropa, estado_actual_atenas)**

Si no se siguen estas indicaciones, **tu código no podrá ser revisado.**

Ejemplo

Público #1:

- **Input:**

```
agatha,106,14;briseida,92,26;aquiles,102,11
300
0.6
pistola
```

- **Output:**

```
Antes de la llegada de Ares, Atenas tenia el estado 300
El odio de este turno es 0.6 y Ares atacara con un(a) pistola
```

```
Ares ha atacado al aldeano agatha y ha quedado con 92 puntos de vida
El aldeano briseida ha probado la inmortalidad y ahora tiene 118 puntos de vida
El estado actual de Atenas es 404
¡Han ganado las tropas de Atenas!
```

Este es el Output e Input completo de los ejemplos presentados en la explicación de las funciones **ataque_ares** y **calcular_estado_atenas**. La `lista_tropa` es resultado de pasar a lista la primera línea del Input, quedando de la forma:

- `['agatha,106,14', 'briseida,92,26', 'aquiles,102,11']`

La primera línea del Output es resultado del mismo flujo del enfrentamiento, el cual señala que el estado inicial de Atenas es de 300. También la segunda línea es resultado del flujo, e indica que para este turno el odio es 0.6 y Ares atacará con una pistola.
Luego, se llama a la función **ataque_ares** y, que imprime las líneas 3 y 4, y a **calcular_estado_atenas**, la cual imprime la quinta línea del Output. Puedes ver más detalles en los ejemplos de cada función en Objetivo.
Como 404 (resultado explicado en el ejemplo de Objetivo) es mayor que el estado original de la ciudad (300), se imprime la última línea del Output, la cual también es resultado del flujo entregado.

Público #2:

- **Input:**

```
evangelina,190,17;alysa,10,15;adara,150,10;adonis,50,12
400
0.8
espada
```

- **Output:**

```
Antes de la llegada de Ares, Atenas tenia el estado 400
El odio de este turno es 0.8 y Ares atacara con un(a) espada
El aldeano evangelina ha probado la inmortalidad y ahora tiene 207 puntos de vida
Se ha reportado el descenso del aldeano alysa
El aldeano adara ha probado la inmortalidad y ahora tiene 160 puntos de vida
El estado actual de Atenas es 539
¡Han ganado las tropas de Atenas!
```

En este ejemplo, la pelea comienza con la siguiente `lista_tropa`, resultado de la primera línea de Input:

- `['evangelina,190,17', 'alysa,10,15', 'adara,150,10', 'adonis,50,12']`

Por otra parte, la segunda línea de Input indica el estado inicial de Atenas, el cual es 400, y las siguientes dos líneas las condiciones del turno: primero el odio del turno (0.8) y luego el arma de Ares (espada).
Se comienza el turno llamando a la función **ataque_ares**. Dentro de esta, se calcula la n cantidad de aldeanos afectados, que resulta 3 al multiplicar el odio del turno con la cantidad de aldeanos en la tropa (0.8 * 4 = 3.2) y redondear a la unidad. Luego, Ares ataca con un puntaje igual a 2 (calculado al entregarle el string 'espada' a la función importada **calcular_puntaje_ares**) a los primeros tres aldeanos:

1. El aldeano `evangelina` obtiene 6 puntos (resultado de entregarle su nombre a la función importada **calcular_puntaje_aldeano**) y gana 17 de vida al tener puntaje mayor que el de Ares
2. El aldeano `alysa` obtiene 0 puntos (resultado de entregarle su nombre a la función importada **calcular_puntaje_aldeano**) y pierde vida al tener puntaje menor que el de Ares. Como posee 10 de vida y le infringen 15 de daño, el aldeano fallece.
3. El aldeano `adara` obtiene 2 puntos (resultado de entregarle su nombre a la función importada **calcular_puntaje_aldeano**) y gana 10 de vida al tener puntaje igual al de Ares

La función **ataque_ares** termina retornando la lista:

- `['evangelina,207,17', 'adara,160,10', 'adonis,50,12']`

Para terminar el turno con el estado de la ciudad actualizado, se llama a la función **calcular_estado_atenas**. Se le entrega la lista recién mencionada y el estado de Atenas actual, que es 400. Dentro de la función, se calcula el promedio de vida de la tropa, que resulta igual a 139 ((207+160+50)/3). Como el promedio es mayor que 100, la función retorna que el estado de Atenas es 539 (400+139).
Como 539 es mayor que el estado original de la ciudad (400), se indica en la última línea del Output que han ganado las tropas de Atenas.

5.

Introducción

Luego del ataque de Ares a Atenas, sufriste graves heridas causadas por el Dios de la guerra. Empiezas a sentir sueño y frío hasta que pierdes la conciencia en el campo de batalla. Recuperas el conocimiento y la cosa está que arde. Sientes que te pinchan la espalda y hay un intenso olor a azufre. Ahuyentas a los pequeños diablillos que te rodean y te das cuenta de la situación. En ese momento te arrepientes de ~~usar tabs en vez de espacios~~ todos los errores que cometiste, pero te dispones a escapar y recuperar tu vida. Te pones de pie y distingues enemigos a lo lejos. Te encuentras en el inframundo y Hades lo sabe.

Objetivo

En esta parte tu misión será escapar del inframundo, para esto te enfrentarás a distintos enemigos y pasarás por lugares en los que recuperarás vida. Si logras sobrevivir a todos los enemigos, te enfrentarás al mismísimo Hades. Deberás implementar la función recursiva **escapar_inframundo** que va a recibir una lista de strings, una posición y una cantidad de vida.

- **escapar_inframundo(lista, posicion, vida):** Esta función tiene como objetivo **simular tu intento de escape del inframundo**, del que puedes salir victorioso, si sobrevives a todos los enemigos o perder contra Hades, si te quedas sin vida.

Esta **función recursiva** recibe tres parámetros: `lista` [*List*], `posicion` [*int*], `vida` [*int*] y retorna un *int* que corresponde a tu vida restante.
En cada recursión deberás sacar (eliminar) un elemento de la **lista** que esté en el índice **posicion** de la lista**,** ese elemento puede ser 2 cosas: un lugar o un enemigo.
Los posibles lugares del inframundo que pueden aparecer son "Tartarus", "Asphodel" y "Elysium". Cualquier otro elemento de la lista que llegues a sacar se asume como enemigo, incluyendo a Hades.
Cada elemento de la lista va a tener entremedio uno o más dígitos. Por ejemplo, C3rbe4rus.
La suma de los dígitos presentes en el elemento visitado van a significar 2 cosas:

1. Cantidad de vida que se pierde (si es un enemigo) o gana (si es un lugar).
2. La posición del siguiente elemento en la lista a visitar.

Mientras tengas vida, deberás seguir sacando (eliminando) elementos de la lista, imprimiendo las frases especificadas en **Output Format** y disminuir la vida como se indica anteriormente. Los enemigos y lugares deben imprimirse sin dígitos entre medio.
Cuando te quedas sin vida, o derrotas a Hades, debes **terminar la recursión y retornar tu vida restante.** En caso de quedarte sin vida debes retornar 0.
Es importante mencionar que luego de enfrentarte a Hades (sacar a Hades de la lista) pueden quedar elementos en la lista.
La explicación de los print están especificados en **Output Format.**

Input format

Tu misión únicamente es crear las funciones. Los inputs de los testcases son simplemente para que puedas visualizar cómo se está evaluando el testcase. **NO DEBES RECIBIR INPUTS.**

Constraints

- **Debes** realizar la función con **recursividad** o se imprimirá 'Recursividad: Falso' y fallarán los testcases.
- Los enemigos en el output format **NO** deben tener dígitos.

Output format

Si sobrevives a un enemigo deberás imprimir:

```
Has derrotado a {enemigo} [{posicion}], perdiendo {vida_perdida} de vida ({vida_restante})
```

En caso de llegar a "Tartarus", "Asphodel" o "Elysium" debes imprimir la siguiente frase:

```
Has superado {lugar} [{posicion}] y recuperado {vida_recuperada} de vida ({vida_actual})
```

Si tu vida llega a ser menor o igual a cero, debes imprimir:

```
Has perdido contra {enemigo} [{posicion}]
```

En el caso de que se te acabe la vida, o que derrotes a Hades, debes terminar la recursión y retornar la vida restante. En caso de quedarte sin vida debes retornar 0.
¡Recuerda tener cuidado con las tildes, mayúsculas y minúsculas!, ya que si el output tiene una mínima diferencia con el esperado será considerado incorrecto.

Importante

Debes ocupar exactamente los **mismos nombres de las funciones** que se explican en Objetivo. Además, las funciones **no deben recibir ni más ni menos parámetros** que los indicados y **deben recibirlos en el mismo orden**.
En este caso, las funciones son las siguientes:

- **escapar_inframundo(lista, posicion, vida)**

Si no se siguen estas indicaciones, **tu código no podrá ser revisado.**

Ejemplos

Público #1:

- **Input:**

```
lista = ['6Rhadamanthus', 'Alect7o', 'Styx4', 'Ha19de3s', 'Ae9acus',
         'Tartaru4s', 'Magae2ra', 'Cerb6eru5s', '7Spart3a', '1Hecate',
         'Eur3yno7mos', 'E3lysium', '1Achlys', 'Asphode0l',
         '19C3orfu7', '5Melinoe', 'Per3sephone', 'Cha3ron',
         'Mi9nos', 'Th4anatos', 'Tisipho2ne', 'Ny0x']
pos = 4
vida = 34
resultado = escapar_inframundo(lista, pos, vida)
if resultado == 0:
    print("No has logrado escapar de Hades")
else:
    print("Has derrotado a Hades y vuelto a Hellas!")

```

- **Output:**

```
Has derrotado a Aeacus [4], perdiendo 9 de vida (25)
Has derrotado a Eurynomos [9], perdiendo 10 de vida (15)
Has derrotado a Achlys [10], perdiendo 1 de vida (14)
Has derrotado a Alecto [1], perdiendo 7 de vida (7)
Has derrotado a Hecate [7], perdiendo 1 de vida (6)
Has derrotado a Styx [1], perdiendo 4 de vida (2)
Has perdido contra Cerberus [4]
No has logrado escapar de Hades
```

En este caso el primer elemento de la lista que se saca es "Ae9acus", porque se encuentra en la posición 4. Se debe limpiar el elemento extrayendo el único dígito que tiene, el cual es 9. Posteriormente se imprime el output correspondiente cuando el elemento es

un **enemigo,** perdiendo 9 puntos de vida. Luego se hace otra llamada recursiva a la posición 9, sin "Ae9acus" en la lista y con la vida restante, que sería 25. Es importante notar que la posición del próximo elemento que se va a extraer de la lista no incluye el elemento que se extrajo anteriormente, así "Eurynomos" que en un principio su posición era 10, luego de la primera recursión su posición ahora es 9 y este es el elemento que debes extraer. Finalmente si se sigue realizando recursiones llegamos al caso base en que nos quedamos sin vida y la recursión termina.

Público #2:

- **Input:**

```
lista = ['1Magaera', 'Elysium3', '1Mete625o4ra', 'Had8es2', 'Achly0s',
         'Eurynomo6s', 'C5haron', 'Minos1', 'N8yx', 'Tart8arus',
         'Thanatos8', '6Styx', 'Asphod5el']
pos = 8
vida = 59
resultado = escapar_inframundo(lista, pos, vida)
if resultado == 0:
    print("No has logrado escapar de Hades")
else:
    print("Has derrotado a Hades y vuelto a Hellas!")
```

- **Output:**

```
Has derrotado a Nyx [8], perdiendo 8 de vida (51)
Has superado Tartarus [8] y recuperado 8 de vida (59)
Has derrotado a Thanatos [8], perdiendo 8 de vida (51)
Has derrotado a Styx [8], perdiendo 6 de vida (45)
Has derrotado a Charon [6], perdiendo 5 de vida (40)
Has derrotado a Eurynomos [5], perdiendo 6 de vida (34)
Has superado Asphodel [6] y recuperado 5 de vida (39)
Has derrotado a Minos [5], perdiendo 1 de vida (38)
Has superado Elysium [1] y recuperado 3 de vida (41)
Has derrotado a Achlys [3], perdiendo 0 de vida (41)
Has derrotado a Magaera [0], perdiendo 1 de vida (40)
Has derrotado a Hades [1], perdiendo 10 de vida (30)
Has derrotado a Hades y vuelto a Hellas!
```

En este caso el primer elemento de la lista que se saca es "N8yx", porque se encuentra en la posición 8. Se debe limpiar el elemento extrayendo los dígitos que suman 8. Posteriormente se imprime el output correspondiente cuando el elemento es un **enemigo,** perdiendo 8 puntos de vida. Luego se hace otra llamada recursiva a la posición 8, sin "N8yx" en la lista y con la vida restante, que sería 51. Es importante notar que la posición del próximo elemento que se va a extraer de la lista no incluye los

elementos que se extrajeron anteriormente, por lo que '5T8arta1rus' que en un principio su posición era 9, luego de la primera recursión su posición es 8, y este es el elemento que debes extraer. Por último si se sigue realizando recursiones llegamos al caso base en que nos enfrentamos a Hades, lo derrotamos y la recursión termina

6.

Introducción

¡Felicitaciones! Has escapado del inframundo y derrotado a Hades en su propio reino. Con todo ello, has impresionado a los dioses Olímpicos y Zeus te traerá a su corte en el Monte Olimpo donde pondrán tu canción favorita, te harán una última prueba de valor y, además, revelarán la identidad de tu padre.
Es entonces cuando los titanes atacan una vez más. Estos estaban encerrados en el Tártaro (parte más profunda del Inframundo) y, por alguna razón desconocida, este quedó con las puertas debilitadas. Estabas en la mitad de tu prueba de valor con los dioses cuando se enteran del inminente ataque al monte Olimpo.
Es ahí cuando Zeus te da su bendición, la cual te da una tremenda fuerza digna de Hércules y una agilidad en combate propia del mismísimo Aquiles para prepararte la primera línea de defensa del Olimpo. Ahí también Poseidón te revela que es tu padre y está muy orgulloso de ti y del héroe/heroína en que te has convertido, así que él también te otorga el poder del tridente de Poseidón dándote una fuerza de combate incalculable.

Objetivo

Para esta parte tu objetivo es detectar cuándo llegan los titanes y luego enfrentarte a ellos uno por uno.
En primer lugar, se te entregará una cantidad **N** de arenas titánicas con inscripciones (`strings`) seguidas por **una** llave (`string`). Tanto las arenas como las llaves son strings compuestos únicamente por X y O. Además, deberás recibir tus puntos de vida iniciales (`int`).
Para cada arena titánica, deberás buscar si la llave está en ella. Si aparece la llave en dicha arena, significa que te has encontrado con un titán y deberás enfrentarte a este.
Al inicio de cada enfrentamiento, recibirás el nombre (`string`) y puntos de vida (`int`) del titán a enfrentar. El combate con un titán puede tener uno o más rondas, lo que dependerá de cuantas veces se encuentre la llave en la arena titánica.
Cada round depende a su vez de la arena en la que se está llevando a cabo la batalla, y sigue el siguiente formato:

Para cada round, es decir, cada vez que se encuentre la llave, debes mostrar el número de round, y luego revisar los símbolos que componen la arena titánica:

- Si el símbolo es **X**, significa que recibes un ataque del titán, el cual siempre daña en 1 punto tus puntos de vida.
- Si el símbolo es **O**, significa que atacas al titán. El daño que realizas sobre el titán se obtiene al pasarle **tus puntos de vida** a la función `calcular_dano_a_titan(vida)`, que puedes importar del módulo **zeus.py**. Una vez obtenido el daño, deberás restar ese número de los puntos de vida del titán.

Esto se repetirá hasta que se acaben los símbolos de la arena titánica (las veces que esté la llave en la arena), o ya sea tú o el titán queden sin puntos de vida. En caso de terminar un enfrentamiento contra un titán deberás reportarlo, junto a la razón del término (ver sección **Output format** para más detalles).
Luego de terminar un enfrentamiento, y en caso de que aún tengas vida, deberás seguir recorriendo la arena en la que te encuentras buscando por la próxima llave.
Finalmente, ya sea si has recorrido todas las arenas invicto/a, o has perdido contra los titanes, deberás reportar el desenlace de la batalla.

Input format

- Recibirás 1 número entero que indicará la cantidad de `strings` a que a continuación vendrán.
- Recibirás tantos `strings` sueltos pertenecientes a arenas titánicas, como indique el número anterior.
- Recibirás 1 `string` extra que tendrá una llave a buscar.
- Recibirás 1 número entero que corresponde a tu vida.
- Por cada `string` que tenga la llave presente recibirás 2 inputs extra correspondientes al nombre de del titán y un número entero de la vida del mismo.

Constraints

- En caso de que el titán o tú se queden sin vida, se termina el enfrentamiento.
- En caso de quedarte sin vida, se debe terminar la batalla.
- Las vidas nunca pueden ser negativas.
- Los `strings` de las arenas titánicas y de la llave siempre tendrán símbolos del tipo X y O.

Output format

Al encontrarte a un titán deberás notificar su encuentro, seguido de una línea en blanco:

```
> Me he encontrado con {nombre_titan}

```

Al inicio de cada ronda del enfrentamiento, deberás mostrar el número de esta, partiendo desde 1. Esto debe estar seguido, nuevamente, de una línea en blanco

```
>> Ronda {numero_de_ronda}
```

```

```

Durante cada ataque de una ronda deberás ir indicando que pasa:
En primer lugar, se imprime el estado de ambos contrincantes antes del ataque:

```
Mi vida actual es {vida_semidios}
La vida de {nombre_titan} es {vida_titan}
```

Y luego, en caso de que ataques tú se debe mostrar:

```
>>> Le cause {Y} de dano a {nombre_titan}

```

Por otro lado, en caso de que ataque el titán, se debe mostrar:

```
>>> {nombre_titan} me ataco y perdi vida

```

cuidado: nota que las dos opciones anteriores están seguidas de una línea en blanco
Al terminar un enfrentamiento con un titán se deberá reportar la razón de este:

Condición	Print
Sigues teniendo vida y el enemigo se quedó sin vida	`{nombre_titan} derrotado con exito!`
Sigues teniendo vida y el enemigo también (la razón de término fue terminar de recorrer el `string`)	`{nombre_titan} ha escapado de ti con {vida_titan} de vida!`
Te quedaste sin vida	`Te ha derrotado {nombre_titan}`

Este mensaje debe estar seguido también de una línea en blanco
Una vez le ganas a todos los titanes o pierdes toda tu vida, lo reportas al final, según corresponda:

```
Has perdido la batalla contra los titanes y las titanides :(
Has ganado la titanomaquia y con ello has ganado el favor de Zeus y todos los olimpicos
```

¡Recuerda tener cuidado con las mayúsculas y minúsculas, y que ningún input ni output llevará tildes ni eñes!

Ejemplos

Público #1

- **Input:**

```
1
XX000X000X
X000
20
Oceano
30
```

El primer input (int) me indica que recibiré 1 arena titánica (string), luego recibo la llave (string) y después mi vida. Como la llave está en 1 string (el único que hay), significa que aparecerá 1 titán. Por ello aparece el malvado titán Océano (rival de tu padre Poseidón) a enfrentarte con 30 de vida.
Ahora bien, como la llave esta 2 veces, deberás recorrer cada símbolo de la arena titánica hasta terminarlas o que alguno de los 2 quede derrotado.

- **Output:**

```
> Me he encontrado con Oceano

>> Ronda 1

Mi vida actual es 20
La vida de Oceano es 30
>>> Oceano me ataco y perdi vida

Mi vida actual es 19
La vida de Oceano es 30
>>> Oceano me ataco y perdi vida

Mi vida actual es 18
La vida de Oceano es 30
>>> Le cause 6 de dano a Oceano

Mi vida actual es 18
La vida de Oceano es 24
>>> Le cause 9 de dano a Oceano

Mi vida actual es 18
La vida de Oceano es 15
>>> Le cause 9 de dano a Oceano

Mi vida actual es 18
La vida de Oceano es 6
>>> Oceano me ataco y perdi vida

Mi vida actual es 17
La vida de Oceano es 6
```

```
>>> Le cause 16 de dano a Oceano

Oceano derrotado con exito!

Has ganado la titanomaquia y con ello has ganado el favor de Zeus y todos los olimpicos
```

Al recorrer la arena titánica que contienen la llave, cuando hay símbolos O atacas tú y en las X ataca el titán. Después de cada turno imprimes el flujo y al final quien gana. En este caso te enfrentaste al titán y no perdiste (le quitaste toda la vida), por ello ganaste el enfrentamiento. Al terminar todos los titanes (solo 1 en este caso), imprimes que has ganado esta batalla final.
En este caso, no fue necesario recorrer nuevamente la arena titánica, ya que ya habías derrotado a Oceano

Público #7

- **Input:**

```
2
OXOXOOXOOX
OXXOOXXOXX
OXO
17
Crio
72
```

Recibimos la cantidad de arenas, luego las arenas y a continuación la llave. Podemos ver como la llave está en 1 arena titánica (la primera), pero que esta se repite 3 veces dentro de ella.
Se te otorgan los puntos de vida en la quinta línea de Input. Luego, vienen los nombres de cada titán con sus puntos de vida respectivos.
Inicias el flujo en orden de aparición.

- **Output:**

```
> Me he encontrado con Crio

>> Ronda 1

Mi vida actual es 17
La vida de Crio es 72
>>> Le cause 8 de dano a Crio

Mi vida actual es 17
```

```
La vida de Crio es 64
>>> Crio me ataco y perdi vida

Mi vida actual es 16
La vida de Crio es 64
>>> Le cause 3 de dano a Crio

Mi vida actual es 16
La vida de Crio es 61
>>> Crio me ataco y perdi vida

Mi vida actual es 15
La vida de Crio es 61
>>> Le cause 6 de dano a Crio

Mi vida actual es 15
La vida de Crio es 55
>>> Le cause 14 de dano a Crio

Mi vida actual es 15
La vida de Crio es 41
>>> Crio me ataco y perdi vida

Mi vida actual es 14
La vida de Crio es 41
>>> Le cause 4 de dano a Crio

Mi vida actual es 14
La vida de Crio es 37
>>> Le cause 3 de dano a Crio

Mi vida actual es 14
La vida de Crio es 34
>>> Crio me ataco y perdi vida

>> Ronda 2

```

- Mi vida actual es 13
- La vida de Crio es 34
- >>> Le cause 6 de dano a Crio
-
- Mi vida actual es 13
- La vida de Crio es 28
- >>> Crio me ataco y perdi vida
-
- Mi vida actual es 12
- La vida de Crio es 28
- >>> Le cause 6 de dano a Crio
-
- Mi vida actual es 12
- La vida de Crio es 22
- >>> Crio me ataco y perdi vida
-
- Mi vida actual es 11
- La vida de Crio es 22
- >>> Le cause 5 de dano a Crio
-
- Mi vida actual es 11
- La vida de Crio es 17
- >>> Le cause 4 de dano a Crio
-
- Mi vida actual es 11
- La vida de Crio es 13
- >>> Crio me ataco y perdi vida
-
- Mi vida actual es 10
- La vida de Crio es 13
- >>> Le cause 5 de dano a Crio
-
- Mi vida actual es 10
- La vida de Crio es 8
- >>> Le cause 4 de dano a Crio
-
- Mi vida actual es 10

```
La vida de Crio es 4
>>> Crio me ataco y perdi vida

>> Ronda 3

Mi vida actual es 9
La vida de Crio es 4
>>> Le cause 5 de dano a Crio

Crio derrotado con exito!

Has ganado la titanomaquia y con ello has ganado el favor de Zeus y todos los olimpicos
```

En este caso, como la primera arena titánica contiene la llave tres veces, existen potencialmente tres rondas. Durante las dos primeras rondas, no se le termina la vida a ninguno de los dos personajes, sin embargo, en el primer turno de la tercera ronda se le termina la vida al titán Crio, por lo que ganas la titanomaquia
Público #12

- **Input:**

```
5
XXOXXXOXOO
XOXXOXOXXO
OOXXOOXOOO
OOOOOXXXXO
OXXOOXOOXO
XOXX
27
Cronos
16
Japeto
40
```

Recibimos la cantidad de arenas, luego las arenas y a continuación la llave. Podemos ver como la llave está en 2 arenas titánicas (las 2 primeras), lo que implica que vendrán 2 titanes. Además, esta se repite en la segunda arena.
Se te otorgan los puntos de vida en la octava línea de Input. Luego, vienen los nombres de cada titán con sus puntos de vida respectivos.
Inicias el flujo en orden de aparición.

- **Output:**

```
> Me he encontrado con Cronos

>> Ronda 1

Mi vida actual es 27
La vida de Cronos es 16
>>> Cronos me ataco y perdi vida

Mi vida actual es 26
La vida de Cronos es 16
>>> Cronos me ataco y perdi vida

Mi vida actual es 25
La vida de Cronos es 16
>>> Le cause 12 de dano a Cronos

Mi vida actual es 25
La vida de Cronos es 4
>>> Cronos me ataco y perdi vida

Mi vida actual es 24
La vida de Cronos es 4
>>> Cronos me ataco y perdi vida

Mi vida actual es 23
La vida de Cronos es 4
>>> Cronos me ataco y perdi vida

Mi vida actual es 22
La vida de Cronos es 4
>>> Le cause 4 de dano a Cronos

Cronos derrotado con exito!

> Me he encontrado con Japeto

>> Ronda 1
```

```
•
•  Mi vida actual es 22
•  La vida de Japeto es 40
•  >>> Japeto me ataco y perdi vida
•
•  Mi vida actual es 21
•  La vida de Japeto es 40
•  >>> Le cause 10 de dano a Japeto
•
•  Mi vida actual es 21
•  La vida de Japeto es 30
•  >>> Japeto me ataco y perdi vida
•
•  Mi vida actual es 20
•  La vida de Japeto es 30
•  >>> Japeto me ataco y perdi vida
•
•  Mi vida actual es 19
•  La vida de Japeto es 30
•  >>> Le cause 4 de dano a Japeto
•
•  Mi vida actual es 19
•  La vida de Japeto es 26
•  >>> Japeto me ataco y perdi vida
•
•  Mi vida actual es 18
•  La vida de Japeto es 26
•  >>> Le cause 18 de dano a Japeto
•
•  Mi vida actual es 18
•  La vida de Japeto es 8
•  >>> Japeto me ataco y perdi vida
•
•  Mi vida actual es 17
•  La vida de Japeto es 8
•  >>> Japeto me ataco y perdi vida
•
```

```
Mi vida actual es 16
La vida de Japeto es 8
>>> Le cause 6 de dano a Japeto

>> Ronda 2

Mi vida actual es 16
La vida de Japeto es 2
>>> Japeto me ataco y perdi vida

Mi vida actual es 15
La vida de Japeto es 2
>>> Le cause 7 de dano a Japeto

Japeto derrotado con exito!

Has ganado la titanomaquia y con ello has ganado el favor de Zeus y todos los olimpicos
```

Al recorrer la arena titánica que contienen la llave, cuando hay símbolos O atacas tú y en las X ataca el titán. Después de cada turno imprimes el flujo y al final quien gana. En este caso te enfrentaste al primer titán y le quitaste toda la vida, por ello ganaste el enfrentamiento. En este caso, hay dos arenas titánicas que tienen la llave, por lo que te enfrentas a dos titanes. El primero, Cronos, es derrotado en una ronda, mientras que el segundo, Japeto, es derrotado en dos rondas.