# Ejercicio 3

Bienvenidos al mundo Pokemon, habitado por unas extrañas criaturas llamadas Pokemon. Los entrenadores de este mundo capturan estos Pokemon para enfrentarse a otros entrenadores. Cómo quieren tener la mayor variedad posible en sus equipos de pokemon, nunca capturan dos pokemon iguales o que estén al mismo nivel. Los entrenadores pueden enfrentar sus pokemons con otros pokemons, o dejar uno de sus pokemons vigilando un gimnasio. Desde el centro de pokemons, que gestiona el mundo Pokemon, tienen instaurada la fiesta del *día de la liberación*, durante la cual todos los entrenadores deben liberar al pokemon que se indique desde el centro.

Se pide implementar un TAD CentroPokemon que implemente las siguientes operaciones:

- `capturar_pokemon (e,p,n)`: Registra al nuevo pokemon `p` con el nivel `n` pasado por parámetro dentro del equipo del entrenador `e`. Si el entrenador no estaba registrado, se registra. Si el entrenador ya había capturado este tipo de pokemon se lanza una excepción del tipo `invalid_argument` con el mensaje `Pokemon repetido`. Si el entrenador ya había capturado a un pokemon de este nivel, se lanza una excepción del tipo `invalid_argument` con el mensaje `Nivel repetido`.

- `vencer_pokemon(e,n)`: El entrenador `e` lucha contra un pokemon de nivel `n`, para ello utiliza el pokemon que tenga capturado con nivel más alto. La operación devuelve el nombre y el nivel del pokemon del entrenador `e` que tenga un nivel más alto. Si el entrenador no está dado de alta en el centro se lanza una excepción de tipo `invalid_argument` con el mensaje `Entrenador no existente`. Si el entrenador no tiene ningún pokemon capturado se lanza una excepción de tipo `invalid_argument` con el mensaje `No tiene pokemons`. Si el pokemon de mayor nivel del entrenador `e` no puede vencer al nivel dado como parámetro por tener un nivel estrictamente menor se lanza una excepción del tipo `invalid_argument` con el mensaje `El pokemon no puede ser vencido`. La función no modifica los pokemons del entrenador, ya que el pokemon sigue estando capturado.

- `dia_de_liberacion(p)`: Todos los entrenadores que tengan un pokemon del tipo indicado por el parámetro `p` deben liberarlo. Esta función devuelve el número total de pokemons liberados durante *el día de la liberación*. La complejidad de la operación puede ser lineal en el número de entrenadores del centro.

- `gimnasio(e)`: El entrenador `e` deja el pokemon más antiguo que tenga, es decir, el que se haya capturado antes, en un gimnasio. El entrenador pierde este pokemon que deja de estar capturado, desapareciendo toda la información que tenga el entrenador sobre él. Si el entrenador no está dado de alta en el centro se lanza una excepción de tipo `invalid_argument` con el mensaje `Entrenador no existente`. Si el entrenador no tiene ningún pokemon para dejar en el gimnasio se lanza una excepción del tipo `invalid_argument` con mensaje `No tiene pokemons`

*Requisitos de implementación.*

La implementación de las operaciones debe ser lo más eficiente posible. Por tanto, debes elegir una representación adecuada para el TAD, implementar las operaciones y justificar la complejidad resultante. Se permite una complejidad lineal en el número de entrenadores en la operación `dia_de_la_liberacion`

Los métodos del TAD no deben mostrar nada por pantalla. El manejo de la entrada y salida de datos se realizará en funciones externas al TAD.

## Entrada

La entrada consta de una serie de casos de prueba. Cada caso está formado por una serie de líneas, en las que se muestran las operaciones a llevar a cabo, una por cada línea: el nombre de la operación seguido de sus argumentos. La palabra `FIN` en una línea indica el final de cada caso.

Los nombres de los entrenadores y pokemons son cadenas de caracteres sin blancos. Los niveles son números enteros positivos menores que $10^9$

## Salida

Para cada caso de prueba se escribirán los datos que se piden. Las operaciones que generan datos de salida son:

- `vencer_pokemon`, escribe *e* `utiliza el pokemon` *p* `para vencer a un nivel` *n*, siendo *e* el nombre del entrenador y *p* el nombre del pokemon.

- `dia_de_liberacion`, escribe `Liberados` *n* `pokemon` *p*, siendo *n* el número de pokemons liberados y *p* el nombre de los pokemons liberados.

- `gimnasio`, escribe *e* `deja el pokemon` *p* `en un gimnasio`, siendo *e* el nombre del entrenador y *p* el nombre del pokemon que se deja en el gimnasio.

Si alguna operación produce una excepción se mostrará el mensaje `ERROR: "` seguido del mensaje de la excepción como resultado de la operación, y nada más.

Cada caso termina con una línea con tres guiones (`---`).

## Entrada de ejemplo

```
capturar_pokemon e1 p1 10
capturar_pokemon e1 p2 20
capturar_pokemon e1 p3 5
capturar_pokemon e2 p1 30
capturar_pokemon e2 p4 7
dia_de_liberacion p1
vencer_pokemon e1 15
gimnasio e1
gimnasio e2
dia_de_liberacion p5
vencer_pokemon e1 5
vencer_pokemon e1 8
capturar_pokemon e2 p6 1
capturar_pokemon e2 p3 5
capturar_pokemon e1 p5 30
capturar_pokemon e1 p6 1
dia_de_liberacion p3
dia_de_liberacion p6
gimnasio e1
gimnasio e1
gimnasio e2
FIN
capturar_pokemon e1 p1 10
capturar_pokemon e2 p1 10
capturar_pokemon e1 p1 15
capturar_pokemon e2 p2 10
gimnasio e3
vencer_pokemon e3 4
gimnasio e3
dia_de_liberacion p1
vencer_pokemon e1 1
FIN
```

## Salida de ejemplo

```
Liberados 2 pokemon p1
e1 utiliza el pokemon p2 con nivel 20 para vencer a un nivel 15
e1 deja el pokemon p2 en un gimnasio
e2 deja el pokemon p4 en un gimnasio
Liberados 0 pokemon p5
e1 utiliza el pokemon p3 con nivel 5 para vencer a un nivel 5
ERROR: El pokemon no puede ser vencido
Liberados 2 pokemon p3
Liberados 2 pokemon p6
e1 deja el pokemon p5 en un gimnasio
ERROR: No tiene pokemons
ERROR: No tiene pokemons
---
ERROR: Pokemon repetido
ERROR: Nivel repetido
ERROR: Entrenador no existente
ERROR: Entrenador no existente
ERROR: Entrenador no existente
Liberados 2 pokemon p1
ERROR: No tiene pokemons
---
```

**Autor:** Isabel Pita