

# A

## Tipo A

En mi barrio hay una academia que ofrece cursos de chino mandarín estándar. La academia dispone de varios grupos, numerados del 1 al 6, que se corresponden con los distintos niveles de los exámenes estandarizados HSK, siendo el nivel 1 el más básico y el nivel 6 el más avanzado. Un estudiante puede comenzar matriculándose en un grupo de la academia, e ir avanzando progresivamente por los grupos superiores restantes. Cuando el estudiante supera el nivel 6, se le considera *graduado*.

La directoría de la academia quiere desarrollar un sistema para gestionar los estudiantes matriculados en sus cursos y aquellos que han finalizado sus estudios. Para ello necesita implementar un TAD `academia_chino` con las siguientes operaciones:

- `nuevo_estudiante(dni, grupo)`: Añade a la academia un estudiante con el `dni` dado (un `string`) y lo matricula en el `grupo` pasado como parámetro (un `int`). Si el estudiante ya estaba previamente en la academia (bien sea como estudiante actualmente matriculado o como estudiante graduado) se lanza una excepción `domain_error` con el mensaje `Estudiante existente`. Si el `grupo` dado no es un número de grupo válido, se lanza una excepción `domain_error` con el mensaje `Grupo incorrecto`.
- `promocionar(dni)`: Elimina al estudiante con el `dni` dado del grupo en el que esté matriculado y lo matricula en el grupo de nivel inmediatamente superior, salvo si el estudiante ya estuviese matriculado en el grupo 6, en cuyo caso el estudiante pasa a considerarse graduado. Si no existe ningún estudiante con el `dni` dado en la academia, se lanzará una excepción `domain_error` con el mensaje `Estudiante no existente`. Si el estudiante ya estaba graduado cuando se hace la llamada a `promocionar()`, se lanza una excepción `domain_error` con el mensaje `Estudiante ya graduado`.
- `grupo_estudiante(dni)`: Devuelve un `int` indicando el grupo en el que está matriculado el estudiante con el `dni` dado. Si el estudiante no existe en la academia se lanzará una excepción `domain_error` con el mensaje `Estudiante no existente`. Si el estudiante ya estaba graduado se lanza una excepción con el mensaje `Estudiante ya graduado`.
- `graduados()`: Devuelve en un tipo de datos lineal ordenado alfabéticamente los DNIs de los estudiantes graduados en la academia.
- `novato(grupo)`: Devuelve el DNI de la persona que más recientemente se ha matriculado en el `grupo` dado (por ser nueva o por promoción) y que aún sigue matriculada en el mismo. Si el `grupo` es incorrecto, se lanzará una excepción `domain_error` con el mensaje `Grupo incorrecto`. Si el `grupo` está vacío, se lanzará una excepción `domain_error` con el mensaje `Grupo vacío` (sin tilde).

### *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.

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.

## Salida

Las operaciones `nuevo_estudiante` y `promocionar` no producen salida, salvo en caso de error. Con respecto a las restantes:

- Tras llamar a `grupo_estudiante` debe imprimirse una línea con el texto `XXX esta en el grupo N`, siendo `XXX` el DNI del estudiante consultado y `N` el grupo en el que está matriculado.
- Tras llamar a `graduados` debe imprimirse una línea con el texto `Lista de graduados:` seguida por los DNIs de los estudiantes graduados, separados por espacios.
- Tras llamar a `novato` debe imprimirse una línea con el texto `Novato de N: XXX`, donde `N` es el número de grupo consultado y `XXX` es el DNI resultante.

Cada caso termina con una línea con tres guiones (`---`). Si una operación produce un error, entonces se escribirá una línea con el mensaje `ERROR:`, seguido del error que devuelve la operación, y no se escribirá nada más para esa operación.

### Entrada de ejemplo

```
nuevo_estudiante 123A 1
grupo_estudiante 123A
promocionar 123A
grupo_estudiante 123A
nuevo_estudiante 456B 2
novato 2
FIN
nuevo_estudiante 789C 6
nuevo_estudiante 123C 5
promocionar 123C
promocionar 789C
promocionar 123C
graduados
grupo_estudiante 123C
FIN
nuevo_estudiante 456D 6
nuevo_estudiante 456D 1
promocionar 456D
promocionar 456D
nuevo_estudiante 456D 1
promocionar 789E
FIN
```

### Salida de ejemplo

```
123A esta en el grupo 1
123A esta en el grupo 2
Novato de 2: 456B
---
Lista de graduados: 123C 789C
ERROR: Estudiante ya graduado
---
ERROR: Estudiante existente
ERROR: Estudiante ya graduado
ERROR: Estudiante existente
ERROR: Estudiante no existente
---
```