Estructuras de datos

Grados de la Facultad de Informática (UCM)

Grupo A

26 de junio de 2023

Normas de realización del examen

1. El examen dura **3 horas**.
2. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`.
3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen.
4. Escribe tu **nombre y apellidos** en el espacio reservado para ello en los ficheros proporcionados.
5. Dispones de un fichero plantilla para cada ejercicio. Debes utilizarlo atendiendo a las instrucciones que se dan en cada fichero.
6. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.
7. Si necesitas consultar la documentación de C++, está disponible en `http://exacrc/cppreference`.

Ejercicio 1 (2 puntos)

Se dice que un elemento de una lista es un pivote, si todos los elementos anteriores a él son menores y todos los elementos posteriores a él, son mayores. Dada la clase genérica `Double_linked_list_ed`, que implementa el TAD lista mediante listas doblemente enlazadas circulares con nodo fantasma, debes añadir un nuevo método `void particion()`, que modifique la lista para convertir en pivote el primer elemento, manteniendo el orden de aparición relativo entre el resto de elementos de la lista. Los elementos iguales al pivote se situarán entre los menores que el pivote y los mayores que el pivote. Por ejemplo, dada la lista l = [5, 20, 5, 4, 10, 7, 5, 49, 3], tras la llamada `l.particion()` se transforma en la lista [4, 3, 5, 5, 5, 20, 10, 7, 49].

Requisitos de implementación.

Para implementar el ejercicio se extenderá la clase `double_linked_list_ed` con un método público: `void particion()`.

Para la implementación del método no pueden crearse, directa o indirectamente, nuevos nodos mediante `new` ni borrar nodos mediante `delete`; han de reutilizarse los nodos de la lista de entrada. Tampoco se permite copiar valores de un nodo a otro. El coste de la operación ha de ser lineal con respecto al número de elementos de la lista.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en dos líneas. En la primera línea se indica el número de elementos de la lista. En la segunda línea se muestran los valores de la lista separados por blancos. Estos valores se introducirán en una lista enlazada según aparecen de izquierda a derecha. La entrada termina con un valor 0 que no debe procesarse.

Los valores son números enteros que pueden almacenarse en una variable de tipo `int`.

Salida

Para cada caso de prueba se escribirá una línea con los valores de la lista una vez aplicada la operación.

Entrada de ejemplo

```
6
5 4 6 3 3 1
9
5 3 4 8 9 7 5 5 4
7
5 5 5 5 5 5 5
9
3 7 4 3 8 9 3 8 2
5
7 8 9 8 7
5
4 2 3 1 2
7
7 8 5 4 3 7 8
0
```

Salida de ejemplo

```
4 3 3 1 5 6
3 4 4 5 5 5 8 9 7
5 5 5 5 5 5 5
2 3 3 3 7 4 8 9 8
7 7 8 9 8
2 3 1 2 4
5 4 3 7 7 8 8
```

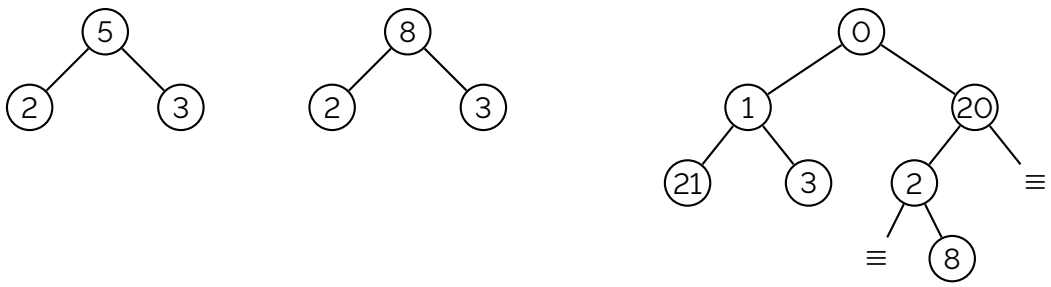
Ejercicio 2 (2 puntos)

Se ha descubierto un mapa del archipiélago *East Blue* que contiene información sobre el número de tesoros que hay en cada una de las islas que componen el archipiélago. El mapa revela que el archipiélago tiene forma de árbol binario, donde los nodos son cada una de las islas, y cuyo valor indica el número de tesoros que se encuentran allí escondidos. Finalmente, las aristas son las corrientes marinas que permiten ir de una isla a otra. Las fuertes corrientes marinas hacen que los barcos solo puedan navegar hacia adelante, dirigiéndose siempre hacia la isla más al norte del archipiélago. Visto sobre el plano, la isla más al norte es la raíz del árbol y se debe navegar siempre desde las hojas hacia la raíz.

El pirata Luffy se encuentra al sur del archipiélago y quiere enviar parte de su flota para recoger todos los tesoros. En cada barco puede cargar un máximo de k tesoros. Como es una zona peligrosa, Luffy quiere enviar el mínimo número de barcos que le permita cargar todos los tesoros. También está interesado en saber cuánto espacio para nuevos tesoros queda en los barcos.

Por ejemplo supongamos que cada barco puede llevar hasta 5 tesoros. En el primer archipiélago Luffy mandará un barco para recoger los dos tesoros de la isla de la izquierda y quedará espacio para 3 tesoros más. Luffy mandará otro barco a recoger los 3 tesoros de la isla de la derecha, quedando espacio en este barco para dos tesoros más. Ambos barcos llegan a la isla que tiene 5 tesoros y recogen tres de ellos en un barco y dos en el otro. Con solo dos barcos Luffi ha conseguido recoger todos los tesoros. Sin embargo, en el segundo archipiélago se tiene que enviar un barco extra para terminar de recoger los últimos tesoros, quedando espacio libre enter los tres barcos para dos tesoros más.

Por último en el tercer archipielago y si se supone que cada barco puede llevar hasta 10 tesoros, se necesitan 3 barcos para llevar los 21 tesoros de la isla de la izquierda, y un barco para llevar los tres tesoros de la isla a su derecha, entre estos 4 barcos pueden llevar el tesoro de la isla que tiene 1 tesoro. En las islas del lado derecho, se necesita un barco para llevar los 8 tesoros de la primera isla, este barco puede llevar también los 2 tesoros de la siguiente isla, y hacen falta dos barcos más para llevar los siguientes 20 tesoros.



Entrada

La entrada comienza con el número de casos que vienen a continuación. Cada caso de prueba consiste en una línea con la descripción de un árbol binario: si el árbol es vacío se representa con un –1; si no, primero aparece su raíz, y a continuación la descripción del hijo izquierdo y después la del hijo derecho, dadas de la misma manera. A continuación se muestra una línea con el número de tesoros que pueden llevar los barcos.

Los valores de los nodos son números enteros positivos menores o iguales a 100. El número de tesoros que puede llevar un barco es un valor entero positivo mayor que cero.

Salida

Para cada caso de prueba, se escribirá una línea con el número de barcos necesarios para recoger todos los tesoros y el número de tesoros que todavía podrían cargarse en los barcos separados por un carácter blanco.

Entrada de ejemplo

```
4
5 2 -1 -1 3 -1 -1
5
8 2 -1 -1 3 -1 -1
5
0 1 21 -1 -1 3 -1 -1 20 2 -1 8 -1 -1 -1
10
21 8 5 -1 -1 3 -1 -1 13 11 4 -1 -1 7 -1 -1 2 -1 -1
5
```

Salida de ejemplo

```
2 0
3 2
7 15
15 1
```

Ejercicio 3 (3 puntos)

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
---
```