Estructuras de Datos y Algoritmos

Grados de la Facultad de Informática (UCM)

Examen SEGUNDO CUATRIMESTRE, 31 de mayo de 2019

Normas de realización del examen

1. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`.

2. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen.

3. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.

4. Del enlace **Material para descargar** dentro del juez puedes descargar un archivo comprimido que contiene material que puedes utilizar para la realización del examen (transparencias de clase, implementación de las estructuras de datos, una plantilla de código fuente y ficheros de texto con los casos de prueba de cada ejercicio del enunciado).

5. Los ficheros con las implementaciones de las estructuras de datos están instalados en el juez, por lo que no es necesario subirlos como parte de tu solución (y conviene no hacerlo).

6. Los ejercicios están identificados con el nombre del tema de la asignatura en el que habrían aparecido si hubieran sido propuestos como ejercicios durante el curso. Para obtener la máxima puntuación, las soluciones deberán seguir los criterios exigidos a los ejercicios de ese tema durante el curso.

7. 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.

8. Al terminar el examen, dirígete al puesto del profesor y rellena con tus datos la hoja de firmas que él tendrá. Muéstrale tu documento de identificación.

Primero resuelve el problema. Entonces, escribe el código.

— John Johnson

Comentar el código es como limpiar el cuarto de baño;
nadie quiere hacerlo, pero el resultado es siempre
una experiencia más agradable para uno mismo y sus invitados.

— Ryan Campbell

Ejercicio 1. Tipos de datos lineales (3 puntos)

Estamos programando un *bot* de Telegram que enviará noticias a sus seguidores y ahora nos toca tratar el escabroso tema de los accidentes de aviación. Cuando se produce uno de estos accidentes, suelen hacerse comentarios del estilo "*Este es el accidente más grave desde febrero de 1995*".

Nos han pasado un listado (ordenado cronológicamente) de accidentes ocurridos en el pasado y queremos estar preparados para que, cuando ocurra el siguiente accidente, podamos producir un comentario como el anterior.

De hecho, para probar esta funcionalidad, queremos saber cuál habría sido el comentario cuando se produjo cada uno de los accidentes conocidos.

Entrada

La entrada está formada por una serie de casos. Cada caso comienza con el número N de accidentes conocidos (un número entre 1 y 250.000). A continuación aparecen N líneas con la descripción de cada uno de ellos: una fecha (con formado `DD/MM/AAAA`) y el número de víctimas en ese accidente. Todas las fechas son distintas y el listado está ordenado cronológicamente de menor a mayor.

Salida

Para cada caso se escribirán N líneas. La i-ésima línea contendrá la fecha del último accidente anterior que tuvo (estrictamente) más víctimas que el accidente i-ésimo. Si no existe tal accidente (en particular, eso ocurre siempre para el primero), se escribirá `NO HAY` en su lugar.

Después de cada caso se escribirá una línea con tres guiones (`---`).

Entrada de ejemplo

```
6
19/12/1990 50
01/02/2000 80
10/05/2001 30
20/10/2005 10
08/07/2007 60
10/07/2007 40
```

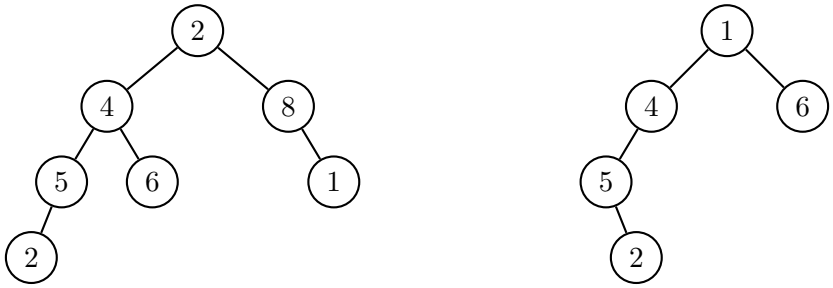
Salida de ejemplo

```
NO HAY
NO HAY
01/02/2000
10/05/2001
01/02/2000
08/07/2007
---
```

Ejercicio 2. Tipos de datos arborescentes (3 puntos)

Definimos un *camino* entre dos nodos de un árbol binario como una secuencia de nodos $n_1 n_2 \ldots n_k$ sin repeticiones (por cada nodo del árbol se pasa como mucho una vez) tal que para todo par de nodos consecutivos $n_i n_{i+1}$ $(1 \leq i < k)$ uno de ellos siempre es padre del otro (n_i es padre de n_{i+1} o n_{i+1} es padre de n_i). Definimos la *longitud* de un camino $n_1 n_2 \ldots n_k$ como el número de nodos que lo forman, k.

Dado un árbol binario de números enteros positivos queremos calcular la longitud del camino más largo formado todo él por números pares. Por ejemplo, el siguiente árbol de la izquierda tiene el camino 6 4 2 8 de longitud 4. En cambio, en el árbol de la derecha todos los caminos formados únicamente por números pares tienen longitud 1.



Entrada

La entrada comienza indicando el número de casos de prueba que vendrán a continuación. Cada caso consiste en una serie de números enteros positivos con la descripción de un árbol binario: el árbol vacío se representa con el valor -1; un árbol no vacío se representa con el valor de un nodo (que denota la raíz), seguido primero de la descripción del hijo izquierdo y después de la descripción del hijo derecho.

Salida

Para cada caso, se escribirá la longitud del camino más largo entre dos nodos del árbol, formado todo él por números pares.

Entrada de ejemplo

```
3
2 4 5 2 -1 -1 -1 6 -1 -1 8 -1 1 -1 -1
1 4 5 -1 2 -1 -1 -1 6 -1 -1
3 4 2 -1 -1 6 -1 8 -1 -1 5 4 -1 2 -1 -1 -1
```

Salida de ejemplo

```
4
1
4
```

Ejercicio 3. Diccionarios (4 puntos)

Tenemos una lista de las películas emitidas durante el último año en una cadena de televisión. De cada película conocemos los actores que intervienen en ella y el tiempo que aparecen en pantalla durante la película. Queremos obtener la película y el actor *preferidos* por la cadena. La película *preferida* es aquella que más veces se ha emitido, mientras que el actor *preferido* es aquel que más minutos aparece en pantalla contando todas las emisiones. Si existen varios actores que han aparecido el mismo tiempo máximo mostraremos todos ellos por orden alfabético. Si existen varias películas que se han emitido el mayor número de veces, mostraremos la que se ha emitido más recientemente de todas ellas.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en varias líneas. En la primera se indica el número P de películas distintas que se han emitido. A continuación se muestra, para cada película, su título y el número A de actores que aparecen en ella. En la línea siguiente se muestra el nombre de cada actor y su tiempo de actuación (un número positivo) en esta película, separados por blancos. A continuación se muestra el número E de películas emitidas por la cadena de televisión en todo el año, seguido de los títulos de las películas en el orden en que fueron emitidas. La entrada finaliza con una línea con cero películas.

Los títulos de las películas y los nombres de los actores son cadenas de caracteres sin espacios en blanco. Los títulos y los nombres de actores están escritos siempre de la misma manera.

Salida

La salida de cada caso se escribirá en dos líneas. En la primera se muestra el máximo número de veces que se ha emitido una película seguido del título de la película. Si existen varias películas que se han emitido el mismo número máximo de veces, se muestra el título de la última emitida. En la segunda línea se muestra el máximo tiempo que ha aparecido un actor en pantalla seguido de los nombres de los actores que han aparecido ese tiempo máximo en orden alfabético.

Entrada de ejemplo

```
2
pelicula1 3
actor1 10 actor2 30 actor3 5
pelicula2 2
actor1 10 actor4 30
3
pelicula2 pelicula2 pelicula1
4
pelicula1 3
actor3 5 actor1 10 actor2 20
pelicula3 1
actor5 40
pelicula4 1
actor1 40
pelicula2 2
actor1 10 actor4 15
5
pelicula2 pelicula1 pelicula2 pelicula3 pelicula1
0
```

Salida de ejemplo

```
2 pelicula2
60 actor4
2 pelicula1
40 actor1 actor2 actor5
```

Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Segundo Cuatrimestre, 30 de mayo de 2019. Grupos B y D

1. (2.5 puntos) "Estremecer" una lista consiste en colocar todos los elementos que aparecen en posiciones pares, por orden de aparición, seguidos de todos los que aparecen en posiciones impares, por orden inverso de aparición (primero el último, después el penúltimo, etc.). Por ejemplo, el resultado de "estremecer" a

0	1	2	3	4	5	6	7

es

0	2	4	6	7	5	3	1

(importante: consideramos que las posiciones comienzan en 0: es decir, la posición del primer elemento es la 0, la del segundo elemento es la 1, etc.)

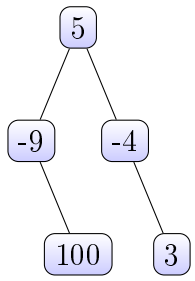
Añade una nueva operación mutadora

```
void estremece();
```

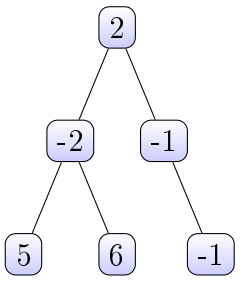
a la implementación del TAD `Lista` basada en nodos doblemente enlazados, que "estremezca" la lista, y determina justificadamente su complejidad. Dicha operación no puede invocar, ni directa ni indirectamente, operaciones de manejo de memoria dinámica (`new`, `delete`), ni tampoco puede realizar asignaciones a los contenidos de los nodos.

2. (2.5 puntos) Un "árbol de ganancias y pérdidas" es un árbol binario de enteros en el que los valores positivos de los nodos representan ganancias, mientras que los negativos representan pérdidas. Un nodo es "rentable" cuando todos sus ancestros son rentables, y, además, la suma de los valores de todos sus ancestros y el del suyo propio es positiva (dicha suma se denomina la "renta" del nodo). El "árbol de ganancias y pérdidas" es "rentable" cuando tiene alguna hoja "rentable" (la "renta" de dicha hoja se denomina una "renta" del árbol).

Ejemplos:



Arbol "rentable"; "renta" máxima: 4



Arbol no "rentable"

Debes programar el procedimiento:

```
void mejor_renta(Arbin<int> a,  bool & es_rentable, int & renta_maxima)
```

que determine: (i) si el "árbol de ganancias y pérdidas" `a` es "rentable" o no; y, en caso positivo, (ii) determine la "renta" máxima del árbol. Si el árbol es "rentable", tras finalizar la ejecución el parámetro `es_rentable` valdrá `true`, y `renta_maxima` contendrá la "renta" máxima del árbol (es decir, la mayor "renta" de las hojas que son rentables). Si el árbol no es "rentable", tras finalizar la ejecución el parámetro `es_rentable` valdrá `false`. En este caso, el valor de `renta_maxima` es irrelevante. También debes determinar justificadamente la complejidad del procedimiento.

3. (5 puntos) La tienda `Outlet aGoGo` nos ha encargado programar un TAD para gestionar su sistema de ofertas online. Este sistema permite ofertar cantidades limitadas de sus productos a precios especiales. Cada vez que un cliente desea beneficiarse de una oferta, el sistema lo pone en la lista de espera para dicha oferta, gestionando las ventas por estricto orden de llegada, hasta que se agoten las unidades disponibles (un cliente podrá esperar simultáneamente para comprar más de un producto). El TAD deberá incluir las siguientes operaciones:

- `crea`: Crea un sistema de venta vacío.
- `an_oferta(producto, num_unidades)`: Crea una nueva oferta para el producto `producto`, con `num_unidades` unidades disponibles. Esta es una operación parcial: no debe existir ya un producto con el mismo nombre en el sistema, y, además, el número de unidades debe ser positivo.
- `pon_en_espera(cliente, producto)`: Añade al cliente `cliente` a la lista de espera para la compra del producto en oferta `producto`. En caso de que el cliente ya esté esperando para comprar dicho producto, la operación no tendrá ningún efecto. Esta es una operación parcial: el producto al que se hace referencia debe estar ofertándose actualmente en el sistema.
- `cancela_espera(cliente, producto)`: Elimina al cliente `cliente` de la lista de espera del producto en oferta `producto`. Si el cliente no está esperando en la lista de espera del producto, la operación no tendrá ningún efecto. Esta es una operación parcial: el producto debe estar ofertándose en el sistema.
- `num_en_espera(producto)`→`num_clientes`: Devuelve el número de clientes que están esperando para comprar el producto `producto`. Esta es una operación parcial: el producto debe estar ofertándose en el sistema.
- `venta(producto, num_unidades)`: Registra una venta de `num_unidades` unidades del producto `producto` al primer cliente de la lista de espera. Dicho cliente se elimina de la lista de espera. En caso de que, tras dicha venta, no queden más unidades, la venta para el producto se cerrará, eliminado la oferta del sistema (y, por tanto, todos los clientes que están esperando se quedarán sin producto). Esta es una operación parcial: el producto debe estar ofertándose actualmente en el sistema, la lista de espera para dicho producto no debe estar vacía, y el número de unidades solicitado no debe sobrepasar el número de unidades disponibles.
- `primero_en_espera(producto)`→`cliente`: Devuelve el nombre del primer cliente que está esperando para comprar el producto `producto`. Esta es una operación parcial: el producto se debe estar ofertando actualmente, y su lista de espera no debe estar vacía.
- `lista_ventas()`→`Lista`. Devuelve una lista del número de unidades vendido para cada producto desde la puesta en marcha del sistema. Cada elemento de esta lista consiste en: (i) el nombre del producto; y (ii) todas las unidades vendidas de este producto (si un producto se ha ofertado varias veces, este valor será el total vendido para cada oferta). La lista estará ordenada alfabéticamente por los nombres de productos.

Aparte de realizar la implementación del TAD, debes indicar justificadamente cuál es la complejidad de cada operación. Al tratarse de un sistema crítico, la implementación deben ser lo más eficientes posible.