Estructuras de Datos

Grados de la Facultad de Informática (UCM)

Convocatoria extraordinaria, 17 de septiembre de 2020

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. 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).

5. 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.

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.

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 (2 puntos)

La intersección de dos listas ordenadas y sin repeticiones es la lista ordenada que contiene los elementos que tienen ambas listas. Por ejemplo, la intersección de la lista `1, 3, 4, 5, 8, 9` con la lista `2, 4, 8, 10` es la lista `4, 8`.

Queremos extender *mediante herencia* la clase `deque<int>`, que implementa las colas dobles de enteros mediante listas de nodos dinámicos, doblemente enlazados, circular y con nodo fantasma, con un nuevo método `interseccion` que recibe como argumento una lista ordenada y modifica la lista de `this` para que termine teniendo la intersección de ambas listas.

En la implementación del nuevo método no pueden hacerse nuevos `new`s, ni *copiar* los enteros de un nodo a otro. También hará falta un método `print` para mostrar por pantalla los elementos de la lista, de inicio a fin.

No modifiques ni subas al juez el fichero `deque_eda.h` cuya clase `deque` debes extender.

Entrada

La entrada consta de una serie de casos de prueba. La primera línea contiene el número de casos de prueba que vendrán a continuación. Cada caso ocupa dos líneas. Cada una de estas líneas representa una de las listas, y contiene sus elementos ordenados de menor a mayor, una serie de números entre 1 y 1.000.000, seguidos de un 0, que marca el final de la descripción de la lista, sin pertenecer a ella. En cada lista, todos sus elementos son distintos.

Salida

Para cada caso de prueba se escribirá en una línea la lista modificada tras hacer la intersección de ambas listas.

Entrada de ejemplo

```
3
1 3 4 5 8 9 0
2 4 8 10 0
1 4 6 0
2 9 0
1 2 3 0
1 2 3 0
```

Salida de ejemplo

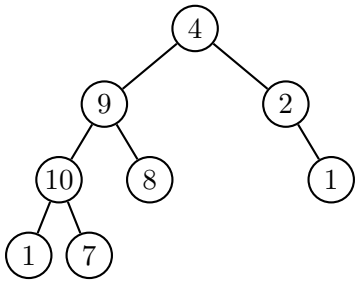
```
4 8

1 2 3
```

Ejercicio 2. Tipos de datos arborescentes (2 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). La *suma* de un camino es la suma de los valores contenidos en los nodos que forman parte de él.

Por ejemplo, supongamos el siguiente árbol binario:



El camino formado por los nodos [1, 10, 9, 4, 2, 1] tiene como suma 27. Por otro lado, el camino formado por los nodos [7, 10, 9, 8] tiene como suma 34. De hecho, este último camino es el que tiene la *suma máxima*, en el sentido en el que no puede obtenerse otra suma mayor con otro camino distinto.

Se implementará una función externa a la clase `bintree` que explore el árbol calculando la suma máxima que puede obtenerse a partir de algún camino del árbol.

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 suma máxima de entre los caminos del árbol.

Entrada de ejemplo

```
3
4 9 10 1 -1 -1 7 -1 -1 8 -1 -1 2 -1 1 -1 -1
2 7 10 1 -1 -1 4 -1 -1 -1 5 -1 -1
2 1 -1 -1 -1
```

Salida de ejemplo

```
34
28
3
```

Ejercicio 3. Diccionarios (3 puntos)

Los aficionados al ciclismo tienen tres grandes acontecimientos en la temporada. En mayo se corre el Giro de Italia, en julio el Tour de Francia y en septiembre La Vuelta a España. Estas carreras se desarrollan por etapas. Al terminar cada etapa se muestra la clasificación de la etapa, que es el tiempo que ha tardado cada ciclista en hacer el recorrido de esa etapa. Por otro lado, tenemos la clasificación general de la carrera, que es el tiempo que ha tardado cada ciclista en recorrer todas las etapas ya transcurridas. Además de la clasificación de los ciclistas, es importante la clasificación de los equipos a los que estos pertenecen. Cada ciclista que participa en la carrera pertenece a un equipo. La clasificación general de los equipos se obtiene sumando el tiempo que han tardado los tres ciclistas del equipo que llegaron primero en cada etapa, siempre que hayan entrado en el tiempo máximo permitido para la misma. Si un equipo queda con menos de tres corredores deja de participar en esta clasificación.

Dependiendo de los kilómetros que se recorren en la etapa y del desnivel que tenga, la organización fija un tiempo máximo para recorrerla. Todos aquellos ciclistas que no llegan a la meta en este tiempo son descalificados y no vuelven a tomar la salida en las siguientes etapas. Por el contrario, suponemos que todos los ciclistas que no son descalificados en una etapa toman la salida en la etapa siguiente. Es decir, ningún ciclista abandona la competición por voluntad propia.

Dada la clasificación de una serie de etapas, la organización quiere que obtengamos tres resultados:

1. El tiempo total de cada equipo que sigue participando en la clasificación por equipos. El listado está ordenado según el orden alfabético de los equipos.

2. El nombre y el tiempo del primer ciclista en la clasificación general de ciclistas (el que menos tiempo acumulado lleva y en caso de empates, el que tiene un nombre menor alfabéticamente).

3. Para cada etapa, un listado de los ciclistas que han sido descalificados en la misma, ordenados de forma alfabética.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea en que se indica el número de equipos que participan en la carrera. A continuación, para cada equipo se da su nombre seguido del número de ciclistas que ha inscrito en la carrera y del nombre de cada uno de estos ciclistas. Cuando termina la información de los equipos se muestra la clasificación de las etapas. Esta comienza con el número de etapas que vienen a continuación. Para cada etapa se muestra en una línea el número de ciclistas que toman la salida y el tiempo máximo permitido en esa etapa y en las líneas siguientes los nombres de los ciclistas y el tiempo que han tardado, ordenados según la clasificación de la etapa.

Los nombres de los equipos y de los ciclistas son cadenas de caracteres sin blancos. En una etapa nunca participa un ciclista que haya sido descalificado anteriormente.

Salida

Para cada caso de prueba se escriben los tres resultados descritos anteriormente, separados por líneas en blanco:

1. En primer lugar, la información relativa a equipos. Para cada uno de ellos se escribe una línea con su nombre seguido del tiempo total invertido por sus ciclistas. Este listado está ordenado alfabéticamente por nombre de equipo. Si no hubiera equipos con al menos tres ciclistas, se escribirá `CLASIFICACION VACIA`.

2. A continuación se muestra la información del primer ciclista clasificado: su nombre y el tiempo total invertido. En caso de empates en el tiempo mínimo, se muestra el que tenga un nombre alfabéticamente menor. Si todos los ciclistas han sido descalificados, se escribirá `NO HAY GANADOR`.

3. Por último se muestra la información de los ciclistas que han sido descalificados en cada etapa: en cada línea se mostrará el número de etapa seguido de los nombres de los ciclistas por orden alfabético y separados por blancos. Si en una etapa no hay ciclistas descalificados, aparecerá solamente el número de etapa en esa línea.

Al terminar la salida de un caso se escribirá una línea con tres guiones, `---`.

Entrada de ejemplo

```
3
Movistar 5  Valverde Mas Soler Cataldo Rojas
JumboVisma 5  Roglic Bennett Dumoulin Gesink Martin
Ineos 5  Bernal Amador Carapaz Castroviejo Rowe
2
15 50
Valverde 10  Roglic 20    Carapaz 20  Soler 20   Mas 30
Rojas 30     Bernal 30    Bennett 30  Rowe 30    Castroviejo 50
Amador 50    Dumoulin 60  Gesink 60   Martin 65  Cataldo 70
11 40
Castroviejo 5  Carapaz 15  Valverde 15  Roglic 15  Mas 20
Rojas 20       Bernal 20   Bennett 30   Rowe 30    Amador 50
Soler 55
1
Movistar 3 Valverde Mas Soler
2
3 14
Valverde 10   Mas 13   Soler 15
2 15
Mas 10   Valverde 11
```

Salida de ejemplo

```
Ineos 120
Movistar 115

Valverde 25

1 Cataldo Dumoulin Gesink Martin
2 Amador Soler
---
CLASIFICACION VACIA

Valverde 21

1 Soler
2
---
```