C01 – Partición de una lista enlazada simple

2 puntos

Partimos de la clase `ListLinkedSingle<T>`, que implementa el TAD lista mediante listas enlazadas simples con nodo fantasma y puntero adicional (`last`) al final de la lista. Queremos añadir un nuevo método, llamado `partition()`:

```
template<typename T>
class ListLinkedSingle {
private:
  struct Node {
    T value;
    Node *next;
  };
  Node *head;
  Node *last;
  int num_elems;

public:
  ...
  void partition(int pivot); // <- Nuevo metodo
};
```

Este método recibe un número entero `pivot` y reorganiza los nodos de la lista enlazada de tal forma que al principio aparezcan aquellos que contengan enteros menores o iguales que el pivote y al final los que contengan enteros mayores que el pivote.

Por ejemplo, si la lista `xs` contiene los valores `[5, 10, 4, 7, 9, 3]` y se realiza la llamada `xs.partition(8)`, la lista `xs` se transforma en `[5, 4, 7, 3, 10, 9]`. Observa que las posiciones relativas entre los elementos de las dos partes es la misma antes y después de la reorganización. Por ejemplo, como el `5` aparece antes que el `4` en la lista original y ambos son menores que el pivote, en la lista resultado aparecen al principio también así: el `5` antes que el `4`.

Importante: 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 las listas de entrada. Tampoco se permite copiar enteros de un nodo a otro. El coste de la operación ha de ser lineal con respecto al tamaño de la lista de entrada.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso se muestra en dos líneas. La primera contiene dos números: el número N de elementos de la lista (un número entre 1 y $100{,}000$) y el valor de `pivot`. En la segunda se muestran los N elementos de la lista, números enteros entre 1 y $1{,}000{,}000$.

Salida

Para cada caso de prueba se escribirá en una línea la lista modificada tras reorganizar sus elementos según el valor del pivote. Para ello se utiliza la representación textual de las listas vista en clase.

Entrada de ejemplo

```
6 8
5 10 4 7 9 3
7 4
1 2 3 4 5 6 7
7 4
7 6 5 4 3 2 1
```

Salida de ejemplo

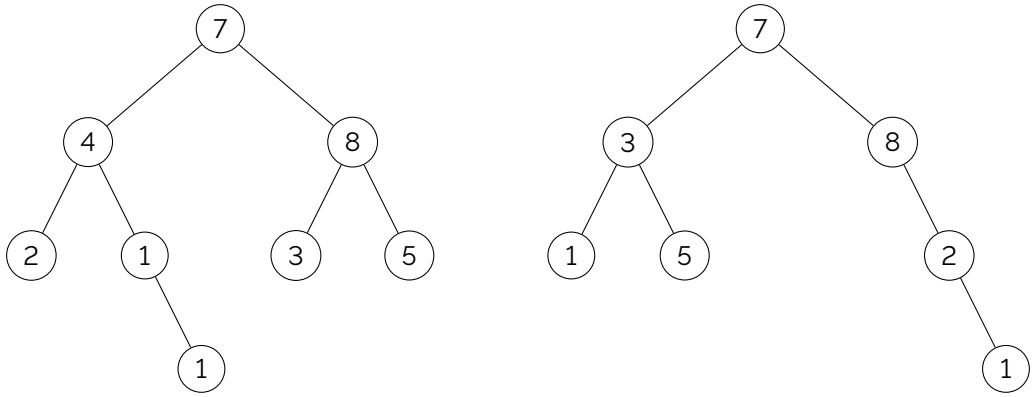
```
[5, 4, 7, 3, 10, 9]
[1, 2, 3, 4, 5, 6, 7]
[4, 3, 2, 1, 7, 6, 5]
```

C02 – Árboles diestros

2 puntos

Dado un árbol binario de enteros, se dice que es *diestro* si o bien es el árbol vacío o una hoja, o bien la suma de los valores de todos los nodos del hijo derecho es mayor que la suma de los valores de todos los nodos de su hijo izquierdo y, además, tanto el hijo izquierdo como el derecho son diestros.

Por ejemplo, de los siguientes árboles, el de la izquierda no es diestro porque el subárbol con raíz 4 no lo es (los descendientes en sus dos hijos suman lo mismo). El de la derecha sí es diestro (todos los nodos cumplen que la suma de los descendientes en su hijo derecho es mayor que la suma de los descendientes en su hijo izquierdo).



En este ejercicio se pide:

1. Definir una función `es_diestro` con la siguiente cabecera:

   ```
   bool es_diestro(const BinTree<int> &tree);
   ```

   Esta función debe explorar el árbol de manera eficiente averiguando si es diestro o no. Puedes definir las funciones auxiliares que necesites.

2. Indicar el coste, en el caso peor, de la función anterior. El coste debe estar expresado en función del número de nodos del árbol de entrada.

Entrada

La entrada comienza con un número que indica el número de casos de prueba que vienen a continuación. Cada caso de prueba consiste en una línea con la descripción de un árbol binario mediante la notación vista en clase. El árbol vacío se representa mediante `.` y el árbol no vacío mediante `(iz x dr)`, siendo `x` la raíz, y `iz`, `dr` las representaciones de ambos hijos.

Salida

Para cada árbol se escribirá una línea con un `SI` si el árbol es diestro y un `NO` si no lo es.

Entrada de ejemplo

```
4
(((. 2 .) 4 (. 1 (. 1 .))) 7 ((. 3 .) 8 (. 5 .)))
(((. 1 .) 3 (. 5 .)) 7 (. 8 (. 2 (. 1 .))))
.
((. 2 .) 5 (. 2 .))
```

Salida de ejemplo

```
NO
SI
SI
NO
```

C03 - Clasificación de *La Vuelta*

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 60  Cataldo 60
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
---
```