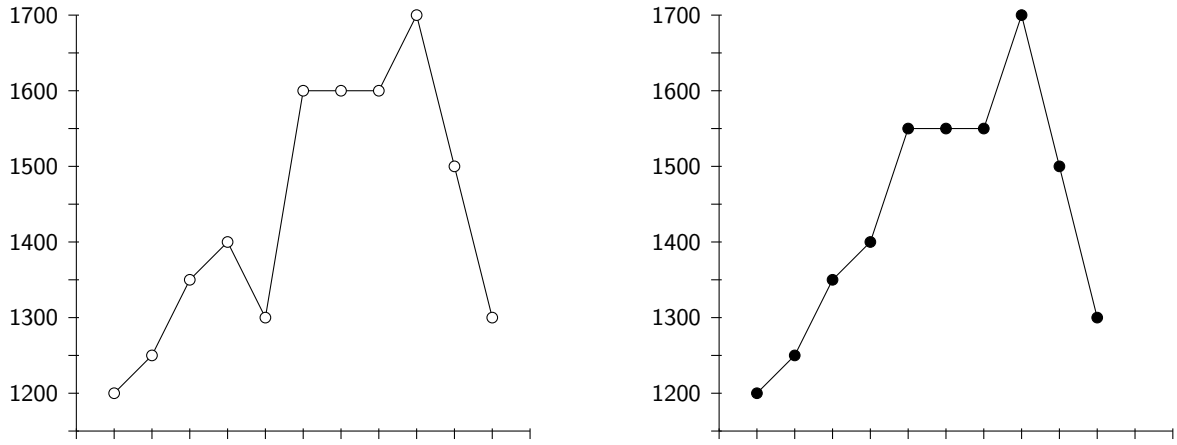
Estructuras de Datos y Algoritmos

Grados de la Facultad de Informática (UCM)

Examen FINAL JUNIO, 1 de junio de 2018

1. (3 puntos) Con la llegada del buen tiempo, el grupo de senderistas "Caminando voy" quiere preparar una serie de excursiones por la Sierra de Madrid. Las excursiones deben ser aptas para todos los socios (hay un numeroso grupo de jubilados y otro de niños de corta edad). En concreto, se requiere que las cuestas arriba no se hagan demasiado penosas. Para ello, para cada excursión se ha confeccionado un perfil de desniveles, consistente en una secuencia de cotas de altura (valores enteros no negativos). Las cuestas arriba se corresponden con segmentos estrictamente crecientes, y el desnivel de una cuesta será la diferencia entre su cota más alta y su cota más baja. Una excursión se considerará *apta* si en todas las cuestas arriba el desnivel no supera un cierto valor estipulado $D \geq 0$.

Por ejemplo, supongamos $D = 300$ metros. Una excursión con el perfil $[1200, 1250, 1350, 1400, 1300, 1600, 1600, 1600, 1700, 1500, 1300]$ sería *apta* (primera gráfica). En cambio, una de perfil $[1200, 1250, 1350, 1400, 1550, 1550, 1550, 1700, 1500, 1300]$ sería *no apta* (segunda gráfica).

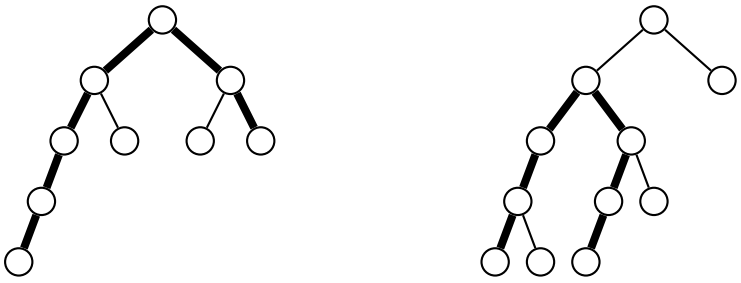


Especifica, diseña e implementa una función que reciba una secuencia no vacía de enteros no negativos (≥ 0) representando el perfil de una excursión, y un entero no negativo correspondiente al desnivel máximo permitido, y determine si la excursión es apta o no.

Escribe el *invariante* y la *función de cota* que permitan demostrar la corrección del algoritmo implementado. Por último, justifica el *coste* del algoritmo conseguido.

El fichero `main1.cpp` contiene código que puedes completar para resolver este ejercicio.

2. (3 puntos) Definimos un *camino* en un árbol binario como una secuencia de nodos $n_1 n_2 \dots n_k$ 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 $n_1 n_2 \dots n_k$ como el número de nodos que lo forman, k. Y definimos el *diámetro* de un árbol como la longitud del camino más largo del árbol. Por ejemplo, los dos árboles siguientes tienen diámetro 7 y un camino de esa longitud aparece resaltado con trazo más grueso en los árboles.



Implementa una función externa a la clase `bintree` que calcule el diámetro de un árbol binario.

El fichero `main2.cpp` contiene ya el código necesario para este ejercicio, salvo la función `diametro`.

3. (4 puntos) Tu tarea consiste en producir la clasificación final de un concurso de programación conociendo los envíos de soluciones a los distintos problemas que han realizado los equipos participantes. Para la clasificación se tienen en cuenta las siguientes reglas:

- El criterio principal para ordenar a los equipos es el número de problemas resueltos. Cuantos más problemas se resuelvan, mejor será el puesto en la clasificación.
- A igualdad de problemas resueltos, los equipos que hayan tardado menos tiempo en resolverlos se clasifican primero.
- El tiempo utilizado por un equipo es igual a la suma del tiempo necesario para resolver cada problema resuelto correctamente.
- El tiempo necesario para resolver un problema es la suma del tiempo transcurrido desde el comienzo del concurso hasta el primer envío correcto a ese problema, más 20 minutos de penalización por cada envío (a ese problema) incorrecto anterior. Los envíos posteriores al primer envío correcto se ignoran.
- Los problemas que finalmente no se resuelven no penalizan.
- Si dos equipos resuelven el mismo número de problemas en el mismo tiempo, serán ordenados por su nombre de menor a mayor.

La entrada consta de una serie de casos de prueba. En la primera línea aparece el número de casos que vendrán a continuación. Cada caso consiste en una serie de descripciones de envíos, cada uno en una línea. Cada envío está formado por el nombre del equipo (una cadena de caracteres sin espacios), el nombre del problema (otra cadena de caracteres sin espacios), los minutos transcurridos desde el comienzo del concurso, y el veredicto del juez (`AC`, `WA`, `TLE`, etc). Un envío se considera correcto solamente si el veredicto es `AC`. Los envíos de cada caso están ordenados de menor a mayor número de minutos. El último envío está seguido de una línea con la palabra `FIN`.

Para cada caso de prueba se escribirá la clasificación final. Esta contendrá una línea por cada equipo que haya realizado algún envío, donde aparecerá el nombre del equipo, el número de problemas resueltos y el tiempo empleado en resolverlos. Los equipos aparecen ordenados según las reglas de clasificación, comenzado por el mejor clasificado. La clasificación de cada caso estará seguida por una línea con "`----`".

El fichero `main3.cpp` contiene código que puedes completar para resolver este ejercicio.

Normas de realización del examen

1. Debes programar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc/domjudge/team`.
2. Escribe comentarios que expliquen tu solución, justifiquen por qué se ha hecho así y ayuden a entenderla. Calcula la complejidad de todas las funciones que implementes.
3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.
4. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que entregues.
5. Descarga el fichero `http://exacrc/Docum122448.zip` que contiene material que debes utilizar para la realización del examen (implementación de las estructuras de datos, ficheros con código fuente y ficheros de texto con algunos casos de prueba de cada ejercicio del enunciado).
6. Si la necesitas, puedes encontrar ayuda sobre la librería estándar de C++ en `http://exacrc/cpp/reference/en/`.
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.