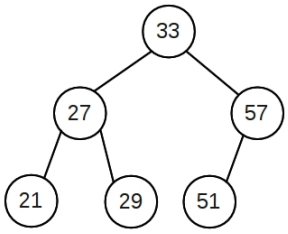
Examen final

1. En clase trabajamos con una implementación de *min-heap* que permitía obtener y borrar el mínimo elemento contenido por la estructura.
Sin embargo, ahora se requiere además de las operaciones ya conocidas, disponer de la posibilidad de encontrar y eliminar el *máximo* elemento existente. Por lo tanto, deberá:

a) Diseñar y explicar un procedimiento que encuentre y retorne el máximo elemento contenido en un *min-heap* (que utilize la implementación vista en clase).
Esta función no podrá recorrer más de la mitad más uno de los elementos del Heap.
NOTA: No hace falta dar una implementación, basta con explicar el algoritmo.

b) Implemente en C la siguiente función `int bheap_erase_max(BHeap *)`, que dado un min-heap binario, remueve el mayor elemento y lo retorna.

2. Un *arbol binario completo ordenado* es un árbol binario completo con la propiedad adicional de que el dato en cada nodo es mayor que los datos en su subárbol izquierdo y menor que los datos en su subárbol derecho. Por ej., el siguiente es un árbol binario completo ordenado:



a) Describir la diferencia entre un árbol binario completo ordenado y un árbol de los utilizados para representar heaps binarios.

b) Sabiendo que los árboles binarios completos se pueden representar como arreglos, implementar una estructura `BSCT` para modelar árboles binarios completos ordenados.

c) Dibujar un árbol binario completo ordenado con elementos 3, 4, 5, 6, 8, 9, 10, 12, 13, y dar la representación del mismo en forma de arreglo.

d) Escribir una fc. `bsct_foreach(BSCT *tree, VisitorFuncInt visit, void *extra_data)` que haga un recorrido "en orden" sobre un árbol binario completo representado en forma de arreglo.

e) Escribir una fc. `BSCT *to_bsctree(int tree[])` que tome un arreglo y devuelva un árbol binario completo ordenado con los elementos del arreglo original. Puede asumir que tiene una función `sort` para ordenar arreglos. Ayuda: al ejecutar un recorrido "en orden" sobre un árbol binario completo ordenado, los elementos se visitan de menor a mayor.

3. El problema del cambio consiste en determinar, dado un número entero C, una combinación de monedas de 1 ctvo., 5 ctvos., 10 ctvos. y 20 ctvos. que sumen C y que use la mínima cantidad de monedas posible.

a) Programar un algoritmo greedy para resolver este problema.

b) ¿Este algoritmo resuelve el problema para cualquier denominación de monedas? Justificar.