Universidad Austral de Chile
Instituto de Informática

**CONTROL SUMATIVO Nº2 – Grupo 1**

*[3.0 puntos]* **PROBLEMA 1**. Ejecución: **./problema1 *n***

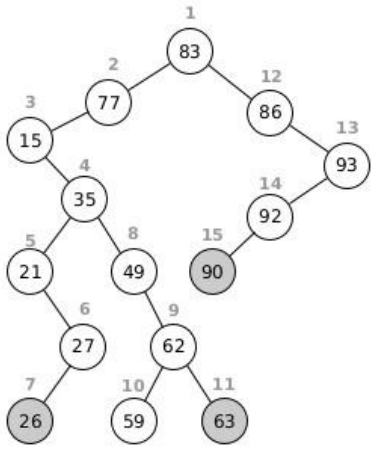
Durante el trabajo en clases se estudiaron los métodos que convierten un arreglo en un *maxheap*. En particular, se trabajó con el método ***makeMaxHeap***, que está disponible en el código fuente ***problema1.cpp***. Utilice dicho método para crear un arreglo de números enteros aleatorios X[1…n] (no considere la posición 0). Luego, debe leer una posición cualquiera del arreglo (valor entero *p*) e intercambiar los valores de los hijos del nodo que se encuentra en esa posición. Para esto, deberá implementar el método **void *intercambiaHijos***(int **X*, int *n*, int *p*); que a partir del valor *p* anterior, valide y reacomode, de ser necesario, los subárboles que tienen como raíz a los hijos de *p*, a fin de que *X* continúe siendo un *maxheap*.

Ejemplo de ejecución para *n* = 30 y *p* = 4:

```
./problema1 30
Arreglo original X[] =
84 87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37 12 69 68 30 83 31 63 24 68 36

heap X[] =
1[94] 2[93] 3[91] 4[87] 5[84] 6[87] 7[78] 8[41] 9[73] 10[37] 11[69] 12[83] 13[63] 14[68] 15[64]
16[16] 17[27] 18[27] 19[50] 20[22] 21[12] 22[63] 23[68] 24[28] 25[30] 26[31] 27[36] 28[24] 29[60]
30[36]

Ingrese posición p: 4

Intercambio en posición p = 4
heap X[] =
1[94] 2[93] 3[91] 4[87] 5[84] 6[87] 7[78] 8[73] 9[50] 10[37] 11[69] 12[83] 13[63] 14[68] 15[64]
16[16] 17[27] 18[27] 19[41] 20[22] 21[12] 22[63] 23[68] 24[28] 25[30] 26[31] 27[36] 28[24] 29[60]
30[36]

Ejecutando test maxheap...
Test maxheap OK
### Fin Problema 1 ###
```

*[3.0 puntos]* **PROBLEMA 2**. Ejecución: **`./problema2 n`**

Se debe crear un BST con valores (claves) aleatorios. Una vez creado, se debe identificar aquellos nodos del árbol que sean "**hojas**" y que, a su vez, tengan una posición (al recorrer en *preorder*) **impar**. En el ejemplo de la figura, las hojas que cumplen esta condición están destacadas en gris:




Ejemplo de ejecución para n = 15:

```
./problema2 15
t en preorder...
83 77 15 35 21 27 26 49 62 59 63 86 93 92 90

Eliminando las claves en v del BST: 26 63 90

t en preorder...
83 77 15 35 21 27 49 62 59 86 93 92
### Fin Problema2 ###
```

Para lo anterior, debe crear un arreglo con las hojas de un BST cuyo preorder sea impar, para después eliminar estas hojas desde el árbol. Para esto:

a) *(0.5 pts)* En el **main**(..), cree un BST, *t*, con *n* claves aleatorias (≤ M ) e imprima el árbol en preorder. Luego cree el vector *v*, vacío y de tipo entero, para invocar al método **encuentraHojas(...)**. A continuación invoque, con *t* y *v* como parámetros, al método **listaEliminando(...)**, para finalizar imprimiendo *t* nuevamente, mostrando que se eliminaron las hojas encontradas.

b) *(2.0 pts)* Cree el método ***void encuentraHojas***(nodoBST **tree*, vector<int> &*v*, int &*pre*); el cual, recursivamente, debe insertar en el vector de enteros *v* las claves de las hojas del árbol cuyo preorder *pre* sea impar.

c) *(0.5 pts)* Cree el método ***void listaEliminando***(nodoBST **tree*, vector¡int¿&*v*); el cual debe imprimir las hojas encontradas y eliminarlas del BST.

***Nota importante****: no necesita modificar nada en la clase BST, utilice los métodos disponibles y trabaje solo en el fuente problema2.cpp.*