

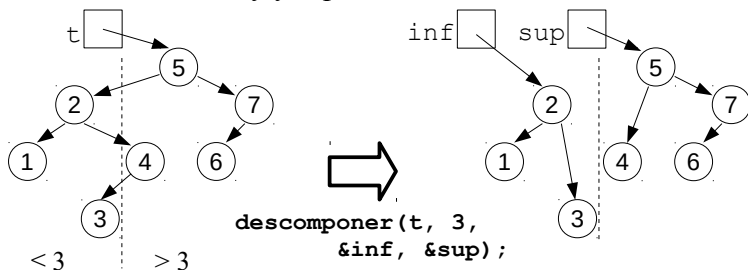
En esta tarea Ud. deberá programar las siguientes funciones:

```
typedef struct nodo {
    int x; // La etiqueta
    struct nodo *izq, *der;
} Nodo;

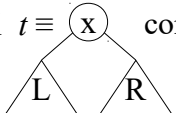
void descomponer(Nodo *t, int z,
                 Nodo **pinf, Nodo **psup);

void descomposicion(Nodo *t, int z,
                    Nodo **pinf, Nodo **psup);
```

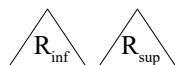
Ambas funciones descomponen el árbol de búsqueda binaria t dejando en $*pinf$ un árbol de búsqueda binaria con los nodos de t con etiquetas menores o iguales a z , y en $*psup$ un árbol de búsqueda binaria con los nodos de t con etiquetas mayores que z . En el siguiente ejemplo de uso, el tipo de las variables t , inf y sup es *Nodo*∗:



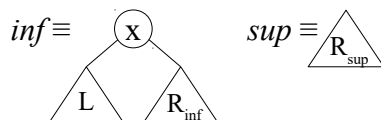
Metodología obligatoria: Si t es el árbol vacío, la solución es trivial.

Si $t \equiv$  con $x \leq z$:

Descomponer recursivamente R, transformándolo en los subárboles:



Luego transformar t entregando como resultado estos 2 árboles:



Cuando $x > z$ se debe descomponer recursivamente L . Deduzca el resto.

Restricciones

- En *descomponer* no pueden usar *malloc*. Deben reutilizar los nodos que recibe en t , modificando los campos *izq* y *der*. No altere x . No se especifica el nodo al cual queda apuntando t .
- En *descomposicion* no puede modificar ninguno de los nodos del

árbol t y por lo tanto debe usar *malloc* para copiar los nodos que necesite modificar. Para cumplir con el requisito de eficiencia, por ejemplo en el caso $x \leq z$, no copie ninguno de los nodos de L , puede reutilizarlos porque no necesita modificarlos. No necesita usar *free*.

- El programa de prueba libera con *free* todos los nodos que Ud. creó y por lo tanto no debe haber *memory leaks* diagnosticados por *sanitiz*.
- Ambas funciones deben ser eficientes, lo que se traduce en que no deben tomar más de un 80% de tiempo de CPU que toman las funciones del profesor.

Instrucciones

Baje *t3.zip* de U-cursos y descomprímalo. El directorio *T3* contiene los archivos (a) *test.c* que prueba si su tarea funciona y compara su eficiencia con la solución del profesor, (b) *prof.ref* con el binario ejecutable de la solución del profesor, (c) *descomponer.h* que incluye el encabezado de las funciones pedidas, y (d) *Makefile* que le servirá para compilar y ejecutar su tarea. Ud. debe programar las funciones *descomponer* y *descomposicion* en el archivo *descomponer.c*.

Ejecute el comando *make* sin parámetros bajo Debian 11. Le explicará qué requisitos debe cumplir para aprobar su tarea, cuáles son las opciones de compilación y ejecución, cómo entregar su tarea, cómo borrar los archivos intermedios y cuál es el trabajo del comando *make*.

Para obtener resultados más consistentes en el benchmark, configure su computador en modo *Alto rendimiento*. Para lograrlo en Windows, posicione el cursor sobre el ícono de la batería, presione el botón derecho y seleccione *Centro de movilidad de Windows*. En *Estado de la batería*, seleccione *Alto rendimiento*.

Entrega

Ud. debe entregar por U-cursos el archivo *descomponer.zip* que genera el comando *make zip*. Recuerde descargar el archivo que subió, descomprimirlo e inspeccionar el contenido para verificar que son los archivos correctos. Se descuenta medio punto por día de atraso, sin considerar recesos, sábados, domingos y festivos. Además se descontará otro medio punto si la compilación arroja *warnings* o no cumple con la indentación requerida en este curso, aunque pase todos los tests.