Parcialito Proyecto 4 - Tema B

Resolver los ejercicios listados abajo, en una hora y media (o menos). El código resultante **no** debe tener memory leaks **ni** accesos (read, write o free) inválidos a la memoria. La entrega es a través de Jaime.

Ejercicio 1

Modificar el TAD `heap` tal que éste provea una función nueva cuya signatura es:

```
heap_elem_t *heap_leaves(heap_t heap)
```

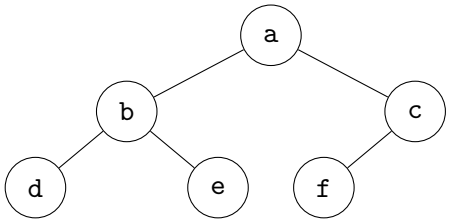
Esta función retorna un arreglo con copias de todas las *hojas* del heap `heap`. Si el heap no tiene hojas, debe retornar `NULL` como resultado.

¿Qué es una hoja del heap? Si pensamos en la implementación que usamos en el taller, sabemos que un heap se representa con el siguiente tipo:

tuple $heap\_t$
  $size$: **nat**
  $elems[1 \dots size]$ **array of** $heap\_elem\_t$
end

Si h tiene tipo $heap\_t$ y $1 \leq i \leq size$, entonces $elems[i]$ es una *hoja* de h si $2 * i > size$, es decir, si la posición i no tiene hijo izquierdo (y por lo tanto no tiene ningún hijo).

Ejemplo En el siguiente heap:



representado por el array `[NULL, a, b, c, d, e, f]` (donde cada elemento es una arista), las hojas son `d`,`e` y `f`.

¿Cuántas hojas tiene un heap? Es fácil ver que con esta representación cualquier heap tiene $(size + 1)/2$ hojas. El heap vacío tiene $1/2 = 0$ hojas. En el ejemplo anterior hay $7/2 = 3$ hojas. Notar que `/` es la división entera, que también existe como operador en C (usando el mismo caracter).

Especificación Las PRE y POST del método `heap_leaves` son las siguientes:

pre El argumento `heap` es un heap de tamaño $size$.

post El resultado es un arreglo de tamaño $(size + 1)/2$, que contiene las hojas de `heap`. Los elementos que tiene el arreglo resultante son una **copia** de los elementos del heap. El arreglo y sus elementos deben ser liberados al terminar de usarse, por el llamador.

Ejercicio 2

Crear un archivo `parcialito.c` que provea una función `main` que haga lo siguiente:

1. Crear un heap vacío de tamaño máximo 10. Imprimir por pantalla las hojas de este heap vacío (**ver abajo** la función auxiliar de impresión para hacer esto).
2. Agregar al heap, una por una, las aristas listadas a continuación. Por cada arista que se agrega, imprimir en pantalla las hojas del heap. Respetar el orden, usar números cualquiera para los vértices, y los siguientes pesos:
   - 8
   - 10
   - 25
   - 23
   - 50
   - 27
   - 105
   - 93
   - 28
   - 74

IMPORTANTE Para mostrar el resultado de cada una de las llamadas arriba listadas, usar la siguiente función auxiliar:

```
void print_result(heap_elem_t * array, unsigned int size) {
    unsigned int i = 0;

    printf("Las hojas son [ ");
    while (i < size) {
        printf("%u ", edge_weight(array[i]));
        i++;
    }
    printf("]\n");
}
```