Ejercicio 1

Explique qué es el **Teorema Maestro**, cómo se aplica y cuándo puede ser aplicado. Explique por qué la complejidad de **Mergesort** puede ser calculada mediante el **Teorema Maestro** pero **Quicksort** no.

Ejercicio 2

Explique cómo funciona **heapsort**.

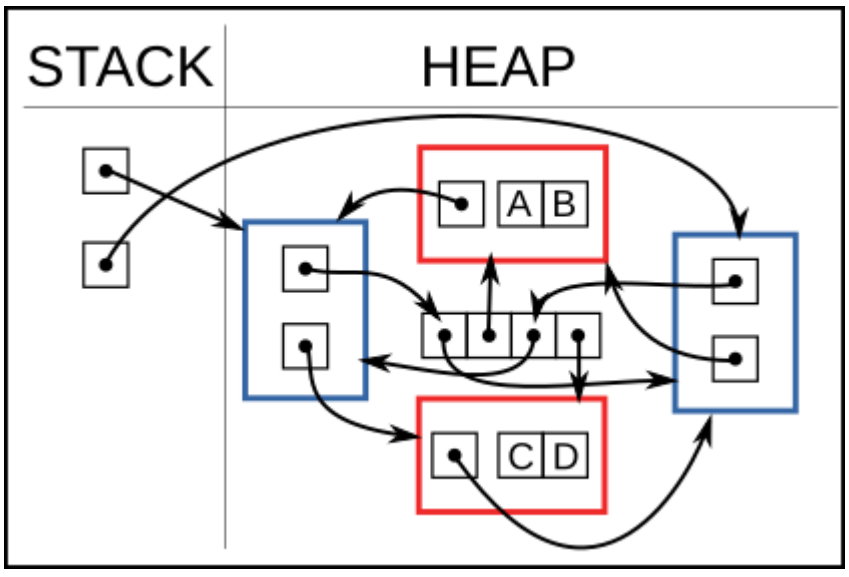
Aplique el algoritmo al vector `V = [6,1,3,6,8,4,2]` de manera que quede ordenado de menor a mayor. Explique cada paso del algoritmo y muestre el estado del vector luego de cada iteración del algoritmo.
Justifique la complejidad del método.

Ejercicio 3

Escriba un programa en `C` que permita armar en memoria las estructuras presentadas en la imagen. Se pueden utilizar variables auxiliares extra en el stack si es necesario.

Agregue también el código para **liberar toda la memoria de forma correcta**. Notese que en el esquema las estructuras se representan con recuardos de colores y los punteros con cuadrados con un punto en el centro. Por otro lado recuerde que los elementos que se encuentran pegados físicamente (horizontal o vertical) representan datos contiguos en memoria, mientras que los elementos separados no.

ATENCION: Este ejercicio evalúa el uso de punteros y memoria dinámica. **Es fundamental que los accesos a memoria sean correctos para que el ejercicio esté aprobado**.



Ejercicio 4

Defina una estructura similiar a la utilizada para el **ABB** pero modificandola para poder implementar un árbol **rojo/negro**. Implemente una función que dado un puntero a esta estructura pueda afirmar si el árbol dado es rojo negro válido o no. Justificar.

Consejo: Puede ser mas fácil (y claro) implementar n funciones (una para cada regla) en vez de intentar hacer todo junto.

Ejercicio 5

Escriba un algoritmo recursivo (sin utilizar **for**, **while**, etc) que dado un string modifique dicho string para eliminar los caracteres en posiciones impares. Por ejemplo:

"" => ""
"A" => "A"
"AB" => "A"
"ABC" => "AC"
"ABCD" => "AC"
"ABCDE" => "ACE"
"ABCDEF" => "ACE"
"ABCDEFG" => "ACEG"

Para ellos se pueden definir solamente dos funciones:

```
char* eliminar_impares(char* str);
void eliminar_impares_recursivo(char* str, int n);
```

No se pueden definir funciones auxiliares (ni utilizar `strlen` y amigos).

Justifique la complejidad de la solución propuesta.