Algoritmos y Estructuras de datos
Ejercicios previos al tercer examen parcial

Los ejercicios que son programas están ordenados por dificultad de menor a mayor

- Resuelve el problema disponible en https://omegaup.com/arena/problem/Uniendo-globos/ sin usar los contenedores de la biblioteca de C++.

- Resuelve el problema disponible en https://omegaup.com/arena/problem/Preorden-de-un-arbol/ sin usar los contenedores de la biblioteca de C++. De todos modos, esta tarea no se resuelve ni con `set` ni con `map`.

- Resuelve el problema disponible en https://omegaup.com/arena/problem/Reubicando-elementos-de-una-list/ sin usar los contenedores de la biblioteca de C++.

- Resuelve el problema disponible en https://omegaup.com/arena/problem/De-una-lista-a-otra/ sin usar los contenedores de la biblioteca de C++.

- Escribe una implementación de la función `cuenta_nodos`, la cual toma un apuntador al nodo raíz de un árbol binario, y devuelve cuántos nodos tiene el árbol. Las hojas (es decir, los nodos sin hijos) apuntan a `nullptr` como sus nodos hijos.

```
struct nodo {
   int valor;
   nodo* izq;
   nodo* der;
};

int cuenta_nodos(nodo* p) {




}
```

- Escribe una implementación de la función `cuenta_apariciones`, la cual toma un apuntador al nodo raíz de un árbol binario además de un entero y devuelve cuántas veces aparece el entero en el árbol. Las inserciones en el árbol se realizaron como sigue: todos los valores del subárbol izquierdo de un nodo son menores o iguales al valor de dicho nodo, mientras que todos los valores del subárbol derecho son mayores. Las hojas (es decir, los nodos sin hijos) apuntan a `nullptr` como sus nodos hijos. Tu función no debe buscar en partes del árbol donde se sabe (por la regla de inserción) que el entero no aparece.

```
struct nodo {
   int valor;
   nodo* izq;
   nodo* der;
};

int cuenta_apariciones(nodo* p, int v) {




}
```