Algoritmos y Estructuras de datos
Ejercicios previos al tercer examen parcial

Problemas de listas enlazadas

- Escribe una implementación de la función `kesimo_valor`, la cual toma un apuntador al nodo inicial una lista enlazada y un entero `k` y devuelve el valor del `k`-ésimo nodo de la lista enlazada (es decir, el valor del nodo al que llegamos después de avanzar `k` veces sobre la lista). Puedes suponer que $k \geq 0$ y que la lista enlazada tiene al menos `k`+1 nodos.

```
struct nodo {
   int valor;
   nodo* sig;
};

int kesimo_valor(nodo* ini, int k) {
   for (int i = 0; i < k; ++i) {
      ini = ini->sig;
   }
   return ini->valor;
}
```

- Escribe una implementación de la función `cuenta_nodos`, la cual toma un apuntador a *algún* nodo válido de una lista enlazada y devuelve cuántos nodos tiene dicha lista. El primer nodo de la lista apunta a `nullptr` como su nodo anterior y el último apunta a `nullptr` como su siguiente nodo.

```
struct nodo {
   nodo* ant;
   nodo* sig;
};

int cuenta_nodos(nodo* p) {
   int res = 0;
   for (nodo* i = p->ant; i != nullptr; i = i->ant) {
      ++res;
   }
   for (nodo* i = p->sig; i != nullptr; i = i->sig) {
      ++res;
   }
   return res + 1;
}
```

- Escribe una implementación de la función `es_circular`, la cual toma un apuntador al primer nodo de una lista enlazada sencilla no vacía y devuelve verdadero si la lista es circular y falso en otro caso. Si la lista es circular, entonces se cumple que el nodo siguiente del último es el primero. Si la lista no es circular, entonces el último nodo apunta a `nullptr` como su siguiente nodo.

```
struct nodo {
   nodo* sig;
};

bool es_circular(nodo* ini) {
   nodo* i = ini->sig;
   while (i != ini && i != nullptr) {
      i = i->sig;
```

```
    }
    return i == ini;
}
```