Algoritmos y Estructuras de datos
Ejercicios previos al primer examen parcial

El apuntador `ini` apunta al primer elemento de la secuencia
El apuntador `fin` apunta después del último elemento de la secuencia

- Escribe una implementación *recursiva* de la función `llena_cero`, la cual toma una secuencia de enteros y la llena toda de ceros. No puedes usar ciclos ni ninguna utilidad de la biblioteca de C++.

```
void llena_cero(int* ini, int* fin) {
   if (ini != fin) {                          // esta implementación es más corta
      *ini = 0;                               // que la vista en clase (en el examen
      llena_cero(ini + 1, fin);               // lo importante es que funcione)
   }
}
```

- Escribe una implementación *recursiva* de la función `es_palindromo`, la cual toma una secuencia de caracteres y devuelve verdadero si la cadena correspondiente es palíndroma y falso en otro caso. No puedes usar ciclos ni ninguna utilidad de la biblioteca de C++.

```
bool es_palindromo(char* ini, char* fin) {       // esta implementación es más corta
   if (fin - ini <= 1) {                         // que la vista en clase (en el examen
      return true;                               // lo importante es que funcione)
   } else {
      return *ini == *(fin - 1) && es_palindromo(ini + 1, fin - 1);
   }
}
```

- Escribe una implementación *recursiva* de la función `es_ordenada`, la cual toma una secuencia de enteros y devuelve verdadero si la secuencia está ordenada de menor a mayor y falso en otro caso. Por ejemplo, la secuencia { 1, 2, 2, 3 } sí está ordenada mientras que la secuencia { 1, 2, 3, 2 } no lo está. No puedes usar ciclos ni ninguna utilidad de la biblioteca de C++.

```
bool es_ordenada(int* ini, int* fin) {           // esta implementación es más corta
   if (fin - ini <= 1) {                         // que la vista en clase (en el examen
      return true;                               // lo importante es que funcione)
   } else {
      return *ini <= *(ini + 1) && es_ordenada(ini + 1, fin);
   }
}
```

- Escribe una implementación *recursiva* de la función `intercambia_parejas`, la cual toma una secuencia de enteros e intercambia los elementos de sus parejas disjuntas contiguas, comenzando por el inicio. Si la secuencia tiene un número impar de elementos, el último elemento no debe modificarse. Por ejemplo, el resultado de aplicar la función a la secuencia { 1, 2, 3, 4, 5, 6 } es { 2, 1, 4, 3, 6, 5 } y el resultado para la secuencia { 1, 2, 3, 4, 5 } es { 2, 1, 4, 3, 5 }. No puedes usar ciclos pero sí cualquier función de C++.

```
void intercambia_parejas(int* ini, int* fin) {
   if (fin - ini <= 1) {
      return;
   } else {
      swap(*ini, *(ini + 1));
      intercambia_parejas(ini + 2, fin);
   }
}
```

- Escribe una implementación *no necesariamente recursiva* de la función `intercambia_mitades`, la cual toma una secuencia de enteros y la modifica intercambiando sus dos mitades. Si la secuencia tiene un número impar de elementos, el elemento de enmedio debe permanecer en su lugar. Por ejemplo, el resultado de aplicar la función a la secuencia { 1, 2, 3, 4 } es { 3, 4, 1, 2 } y el resultado para la secuencia { 1, 2, 0, 3, 4 } es { 3, 4, 0, 1, 2 }. Puedes usar únicamente las utilidades de `algorithm` de la biblioteca de C++.

```
void intercambia_mitades(int* ini, int* fin) {
   int n = fin - ini, m = n / 2 + n % 2;
   for (int i = 0; i < n / 2; ++i) {
      swap(*(ini + i), *(ini + m + i));
   }
}
```