Algoritmos y Estructuras de datos
Ejercicios previos al segundo 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 función que tome una secuencia de enteros y la ordene. No es necesario implementar un algoritmo eficiente, basta con que funcione. Puede usar cualquier utilidad de la biblioteca de C++ excepto `std::sort` y `std::stable_sort`.

```
void ordena(int* ini, int* fin) {
   for (int* i = ini; i != fin; ++i) {
      std::swap(*i, *std::min_element(i, fin));
   }
}
```

- Escribe una función que tome dos alumnos y pueda ser usada como predicado de orden para ordenar alumnos por calificación de mayor a menor; si hay alumnos que tienen la misma calificación, éstos deben ordenarse por número de lista de menor a mayor.

```
struct alumno {
   int lista, calif;
};

bool predicado(alumno a, alumno b) {
   return a.calif > b.calif || (a.calif == b.calif && a.lista < b.lista);
}
```

- Escribe una función que tome un entero positivo N y que devuelva un arreglo que contenga los enteros del 0 al N-1. El arreglo devuelto debe seguir siendo válido aún después de que la función termine. No se permite usar variables globales ni estáticas.

```
int* genera(int n) {
   int* p = new int[n];
   for (int i = 0; i < n; ++i) {                  // se puede hacer más corto
      p[i] = i;
   }
   return p;
}
```

- Escribe una función que tome una doble cola de enteros y devuelva otra con los elementos de la primera pero en orden contrario. No se permite modificar la primera doble cola ni usar las utilidades de `algorithm` de la biblioteca de C++, pero sí las de `deque`.

```
std::deque<int> invierte(const std::deque<int>& d) {
   std::deque<int> res;
   for (int i = 0; i < d.size( ); ++i) {          // se puede hacer más corto
      res.push_front(d[i]);
   }
   return res;
}
```