Algoritmos y Estructuras de datos
Posibles soluciones a los ejercicios previos al segundo examen parcial

En el examen no importa si sus soluciones son diferentes, siempre y cuando funcionen.

- 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* p = ini; p != fin; ++p) {
      std::swap(*p, *std::min_element(p, fin));
   }
}
```

- Escribe una función que tome dos enteros y pueda ser usada por `std::sort` como predicado para ordenar enteros de la siguiente forma: los enteros negativos deben quedar primero y después deben quedar los no negativos, desempatando enteros del mismo grupo por valor absoluto de menor a mayor. Por ejemplo, usar esta función con `std::sort` sobre la secuencia { 8, -7, 1, -2, 6, -4, 0 } debe resultar en { -2, -4, -7, 0, 1, 6, 8 }. Puedes usar cualquier utilidad de la biblioteca de C++.

```
bool signo_magnitud(int a, int b) {
   if (a < 0 && b < 0 || a >= 0 && b >= 0) {
      return abs(a) < abs(b);
   } else {
      return a < b;
   }
}
```

- Escribe una función que tome dos alumnos y pueda ser usada por `std::sort` como predicado 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) {
      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) {
       res.push_front(d[i]);
   }
   return res;
}
```