Algoritmos y Estructuras de datos
Ejercicios previos al segundo examen parcial

Problemas de secuencias dinámicas en dobles colas

- Escribe una función que tome una doble cola de enteros y devuelva otra con una copia de los elementos de la doble cola original, pero que aparezcan el orden contrario. No se permite modificar doble cola original 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;
}
```

- Escribe una función que tome una doble cola y un entero n y rote los elementos de la doble cola n veces hacia la izquierda. Por ejemplo, si la doble cola guarda la secuencia { 1, 2, 3, 4, 5 } y n=2 entonces la doble cola debe quedar { 3, 4, 5, 1, 2 }. No se permite usar las utilidades de `algorithm` de la biblioteca de C++, pero sí las de `deque`.

```
void rota_izquierda(std::deque<int>& d, int n) {
   for (int i = 0; i < n; ++i) {
      d.push_back(d.front( ));
      d.pop_front( );
   }
}
```

- Escribe la implementación de una función que toma dos dobles colas e intente modificar la primera mediante rotaciones para que coincida con la segunda. Si es posible hacerlas coincidir, la función debe realizar la modificación y debe devolver verdadero; en caso contrario, la doble cola no debe presentar modificaciones al término de la función y se debe devolver falso. Por ejemplo, si las dobles colas guardan las secuencias { 1, 2, 3 } y { 2, 3, 1 } respectivamente, entonces la primera doble cola sí puede modificarse para que quede igual que la segunda. No se permite usar las utilidades de `algorithm` de la biblioteca de C++, pero sí las de `deque`.

```
bool intenta_rotacion(std::deque<int>& s1, const std::deque<int>& s2) {
   for (int i = 0; i < s1.size( ); ++i) {
      if (s1 == s2) {       // tanto std::vector como std::deque y std::string
         return true;       // se pueden comparar directamente con ==
      }
      s1.push_back(s1.front( ));
      s1.pop_front( );
   }
   return false;
}
```