Algoritmos y Estructuras de datos
2018-O CSI01
Examen de reposición – Parte 1

Nombre del alumno:
Usuario en OmegaUp: **Calificación máxima: 10**

1. (7.5 puntos) Escribe una implementación de la función `encuentra_ultimo`, la cual toma una secuencia de enteros, un entero `v` y devuelve el apuntador a la última ocurrencia de `v` en la secuencia o el fin de la misma si `v` no aparece. No tienes disponible ninguna utilidad de la biblioteca de C++.

```
int* encuentra_ultimo(int* ini, int* fin, int v) {




}
```

2. (7.5 puntos) Escribe una implementación recursiva de la función `cuenta_digitos`, la cual toma un entero no negativo y devuelve la cantidad de dígitos del mismo. No tienes disponible ninguna utilidad de la biblioteca de C++.

```
int cuenta_digitos(int n) {




}
```

Algoritmos y Estructuras de datos
2018-O CSI01
Examen de reposición – Parte 2

Nombre del alumno:
Usuario en OmegaUp: **Calificación máxima: 10**

3. (7.5 puntos) Escribe una implementación de la función `comparacion_lexicografica` que pueda ser usada por `std::sort` como predicado para ordenar vectores de enteros. Los vectores deben quedar ordenados lexicográficamente de menor a mayor (el orden lexicográfico es el orden del diccionario: se compara componente a componente hasta desempatar y en caso de que un vector se acabe antes de lograr desempatar, el vector más corto es menor). No tienes disponible ninguna utilidad de la biblioteca de C++ a excepción de `vector` y `std::min`.

```
bool comparacion_lexicografica(const std::vector<int>& v1,
                               const std::vector<int>& v2) {









}
```

4. (7.5 puntos) Escribe la implementación de una función `intenta_rotacion`, la cual toma dos dobles colas e intenta 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 secuencia no debe presentar modificaciones al terminar la función y se debe devolver falso. Por ejemplo, si las secuencias son { 1, 2, 3 } y { 2, 3, 1 } entonces la primera secuencia puede modificarse para que quede { 2, 3, 1 }. Puedes asumir que tienes disponibles únicamente las utilidades de deque de la biblioteca de C++.

```
bool intenta_rotacion(std::deque<int>& d1, const std::deque<int>& d2) {
   // Pista: use s1 == s2 para preguntar si las secuencias almacenadas
   // en s1 y s2 son iguales









}
```

Algoritmos y Estructuras de datos
2018-O CSI01
Examen de reposición – Parte 3

Nombre del alumno:
Usuario en OmegaUp: **Calificación máxima: 10**

5. (7.5 puntos) Escribe una implementación de la función `cuenta_nodos`, la cual toma un apuntador a algún nodo 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) {









}
```

6. (7.5 puntos) Escribe una implementación de la función `cuenta_hojas`, la cual toma un apuntador al nodo raíz de un árbol binario y devuelve la cantidad de hojas de dicho árbol. Las hojas (es decir, los nodos sin hijos) apuntan a `nullptr` como sus nodos hijos.

```
struct nodo {
   nodo* izq;
   nodo* der;
};
int cuenta_hojas(nodo* p) {









}
```