PARTE 2 - Equipos: Ejercicios de pseudocódigo
Duración: 60 minutos

EJERCICIO 1

En el corazón de la ciudad inteligente de Algorville, donde la tecnología y la conectividad son fundamentales para la vida cotidiana, se plantea un desafío crucial relacionado con la evaluación de la conectividad en su vasta red de dispositivos interconectados. La municipalidad, en su búsqueda por optimizar la eficiencia de la comunicación en esta Smart City, busca determinar no sólo la existencia de una conexión directa, sino también la posibilidad de conectividad a través de saltos, permitiendo así una comunicación eficaz entre un Nodo X y un Nodo Y.

La red de dispositivos en Algorville está en constante expansión para abordar las demandas de una ciudad inteligente. Los participantes deben desarrollar un algoritmo innovador que no sólo evalúe la conectividad entre nodos específicos, sino que también considere la capacidad de establecer conexiones a través de nodos intermedios, facilitando así una comunicación eficiente en esta infraestructura tecnológicamente avanzada.

En el contexto de una Smart City, donde la interconexión de dispositivos es esencial para optimizar servicios públicos, gestión de tráfico, y eficiencia energética, la solución propuesta deberá adaptarse dinámicamente a las complejidades de la red. Esto incluye enlaces intermitentes, dispositivos temporalmente fuera de servicio y cambios en la topología de la red, elementos que son fundamentales en el desarrollo de una ciudad inteligente.

Se solicita:

- Determinar la estructura de datos apropiada para representar la red de dispositivos de la ciudad.
- Desarrollar un algoritmo que permita determinar si existe una conexión entre 2 nodos y que cumpla con la siguiente firma:

```
Boolean TCiudad.conectados(TVertice nodoX, TVertice nodoY);
```

CONSIDERACIONES IMPORTANTES

- **La estructura propuesta debe buscar optimizar el consumo de memoria de la solución.**
- **Se debe procurar el desarrollo de un algoritmo lo más eficiente posible.**
- Deben respetarse estrictamente las firmas indicadas
- No es necesario desarrollar los métodos de los TDAs utilizados si estos corresponden a las variantes vistas en clase.
- Se deben cumplir todos los pasos estándar de desarrollo de algoritmos, Lenguaje Natural, Pre y Post Condiciones, Pseudocódigo y Análisis del Orden del Tiempo de Ejecución.

EJERCICIO 2

Estás trabajando en un dispositivo IoT que adquiere un gran volumen de datos durante un período específico de tiempo. Por motivos de privacidad, los datos deben ser procesados exclusivamente en el dispositivo (procesamiento on-device). Los datos tienen la siguiente estructura:

TDato:

double medición

int secuencia

Se necesita desarrollar un algoritmo de ordenamiento eficiente para procesar estos datos en el dispositivo, ordenando los datos **según el campo medición en forma descendente**.

Restricciones:

- Memoria limitada en el dispositivo IoT.
- Complejidad temporal eficiente; minimizar el uso de memoria es crucial.
- Los datos se deben ordenar de manera descendente.

TDispositivo:

TDatos[] datos;

void ordenarDatos();

CONSIDERACIONES IMPORTANTES

- **La estructura propuesta debe buscar optimizar el consumo de memoria de la solución.**
- **Se debe procurar el desarrollo de un algoritmo lo más eficiente posible.**
- Deben respetarse estrictamente las firmas indicadas
- No es necesario desarrollar los métodos de los TDAs utilizados si estos corresponden a las variantes vistas en clase.
- Se deben cumplir todos los pasos estándar de desarrollo de algoritmos, Lenguaje Natural, Pre y Post Condiciones, Pseudocódigo y Análisis del Orden del Tiempo de Ejecución