Trabajo Práctico N°2

Objetivos

- Aplicar conceptos teóricos sobre estructuras jerárquicas, grafos y sus algoritmos asociados.

Consignas

1. Sala de Emergencias

El **triaje** es un proceso que permite una gestión del riesgo clínico para poder manejar adecuadamente y con seguridad los flujos de pacientes cuando la demanda y las necesidades clínicas superan a los recursos. El siguiente proyecto de software posee una sencilla simulación de esta situación: enlace a código.
En este proyecto para simplificar existen pacientes con tres niveles de riesgo para su salud: 1: crítico, 2: moderado, 3: bajo. Actualmente la simulación muestra que los pacientes se atienden según el orden en que llegan al centro de salud.

Se solicita:

a. Seleccionar, programar y aplicar una estructura de datos adecuada para almacenar los pacientes conforme ingresan al centro de salud **de modo tal que cuando se atiende un paciente siempre sea aquel cuyo nivel de riesgo es el más delicado** en comparación con el resto de los pacientes que restan por ser atendidos. Si dos pacientes poseen el mismo nivel de riesgo, adoptar un segundo criterio para seleccionar uno de ellos. Recordar que la estructura de datos debe ser genérica, es decir, debe poder almacenar cualquier tipo de dato, y no ser específica para alojar pacientes (separar implementación de aplicación).
b. Fundamentar en el informe, en no más de una página, la estructura seleccionada indicando el orden de complejidad *O* de inserciones y de eliminaciones en la estructura seleccionada.

Observación: Pueden realizarse todas las modificaciones al código que crean necesarias siempre que se respete el propósito del problema planteado.

2. Temperaturas_DB

Kevin Kelvin es un científico que estudia el clima y, como parte de su investigación, debe consultar frecuentemente en una base de datos la temperatura del planeta tierra dentro de un rango de fechas. A su vez, el conjunto de medidas crece conforme el científico registra y agrega **mediciones** a la base de datos.
Una medición está conformada por el valor de temperatura en ºC (flotante) y la fecha de registro, la cual deberá ser ingresada como “dd/mm/aaaa” (string). (Internamente sugerimos emplear objetos de tipo *datetime*).
Ayude a Kevin a realizar sus consultas eficientemente implementando una base de datos en memoria principal “**Temperaturas_DB**” que utilice internamente un árbol AVL. La base de datos debe permitir realizar las siguientes operaciones (interfaz de Temperaturas_DB):

guardar_temperatura(temperatura, fecha): guarda la medida de temperatura asociada a la fecha.

devolver_temperatura(fecha): devuelve la medida de temperatura en la fecha determinada.

max_temp_rango(fecha1, fecha2): devuelve la temperatura máxima entre los rangos fecha1 y fecha2 inclusive (fecha1 < fecha2). Esto no implica que los intervalos del rango deban ser fechas incluidas previamente en el árbol.

min_temp_rango(fecha1, fecha2): devuelve la temperatura mínima entre los rangos fecha1 y fecha2 inclusive (fecha1 < fecha2). Esto no implica que los intervalos del rango deban ser fechas incluidas previamente en el árbol.

temp_extremos_rango(fecha1, fecha2): devuelve la temperatura mínima y máxima entre los rangos fecha1 y fecha2 inclusive (fecha1 < fecha2).

borrar_temperatura(fecha): recibe una fecha y elimina del árbol la medición correspondiente a esa fecha.

devolver_temperaturas(fecha1, fecha2): devuelve un listado de las mediciones de temperatura en el rango recibido por parámetro con el formato “dd/mm/aaaa: temperatura ºC”, ordenado por fechas.

cantidad_muestras(): devuelve la cantidad de muestras de la BD.

Adicionalmente realizar las siguientes actividades:

- Escriba una tabla con el análisis del orden de complejidad Big-O para cada uno de los métodos implementados para su clase “Temperaturas_DB”, explique brevemente el análisis de los mismos.
- Implementar un método para leer un archivo de muestras y cargar a la base de datos de temperaturas para probar sus funcionalidades

3. Palomas mensajeras

Desde la antigüedad y hasta la invención del telégrafo en el año 1835 por Morse, las palomas mensajeras fueron el sistema de comunicación más rápido entre ciudades. La premisa que sustenta este tipo de comunicación es que una paloma mensajera criada en un lugar y liberada en otro, volverá a su lugar de origen; si se aprovecha este viaje haciendo que el animal lleve consigo un mensaje, la información se podrá replicar y transmitir de un lugar a otro.

William, como dueño de la agencia de noticias “Palomas William”, conoce el tema y entiende que hay varios caminos para entregar un mensaje a un lugar. Sin embargo, todavía no está seguro de cómo encontrar la mejor forma de enviar las noticias a las demás aldeas ahorrando recursos. Él tiene a cargo toda la comunicación desde su aldea “Peligros” a un grupo de otras 21 aldeas, todas con un palomar propio. Cada aldea, incluída Peligros, sólo puede enviar mensajes a sus aldeas vecinas de las cuales tiene alguna paloma mensajera. La lista de aldeas vecinas y la distancia entre cada par de ellas está disponible en el archivo aldeas.txt. Cada ruta en la lista es una tupla (S_i, T_i, d_i) donde S_i y T_i son, respectivamente, los nombres (string) de la aldea de inicio y final de cada posible viaje de una paloma mensajera, y d_i representa la distancia en leguas entre esas aldeas (entero positivo).

Ayude a William a encontrar la forma más eficiente de llevar un mensaje desde su aldea a todas las demás, donde cada aldea reciba solamente una vez la noticia. Cada aldea, al recibir la noticia, puede replicarla y enviarla a tantas aldeas vecinas como quiera.

Entregables:

- Mostrar la lista de aldeas en orden alfabético.
- Para cada aldea, mostrar de qué vecina debería recibir la noticia, y a qué vecinas debería enviar réplicas, siendo que se está enviando el mensaje de la forma más eficiente a las 21 aldeas. Tomar en cuenta que desde Peligros solamente se envían noticias a una o más aldeas vecinas.
- Para el envío de una noticia, mostrar la suma de todas las distancias recorridas por todas las palomas enviadas desde cada palomar.

Entregables

Al finalizar el TP, se espera que usted entregue:

- Carpeta de los proyectos (con los ejercicios 1, 2 y 3): separar el código en módulos y una aplicación principal. **Documentar el código, utilizar excepciones para la validación de datos e implementación mediante clases siempre que se considere necesario.**
- Documento (PDF) con una breve explicación de la solución y resultados (conclusiones, gráficas, etc). Puede agregarse algún pseudocódigo si se considera necesario para explicar las soluciones entregadas.
- Debe subirse tanto la carpeta con los proyectos como el PDF a un repositorio github compartido a los docentes de la cátedra. Se tomará como entrega la confirmación (*commit*) correspondiente a la fecha y hora inmediata anterior a la fecha de entrega.