- Los grupos deberán tener exactamente tres (3) integrantes. En caso de que la cantidad total de estudiantes **no** sea múltiplo de 3, se admitirá excepcionalmente algún grupo de dos (2) integrantes.
- Fecha de entrega: ***Martes 16 de Julio de 2024*** hasta las 23:59 hs. Se debe entregar:
  - ❖ Un documento en formato PDF conteniendo:
    - Modelado en términos de un grafo de aquel aspecto del problema que lo amerite.
    - Análisis de tipos abstractos de datos para el problema, justificando la elección.
    - Encabezados de las primitivas y otras operaciones de cada uno de los T.A.D anteriores a utilizar en la implementación, incluyendo toda precondición que sea necesaria.
    - Elección de estructuras de datos para los T.A.D anteriores, justificando la elección y escritura en C++ de los tipos correspondientes (`typedef`).
    - Diagrama de módulos para el sistema, con sus correspondientes inclusiones.
  - ❖ Un archivo **Obligatorio.zip** conteniendo todos los módulos implementados (archivos **.h** y **.cpp**) Se deben **excluir** los archivos compilados (**.o**) y el archivo ejecutable (**.exe**).
- Ambas partes de la entrega deberán estar debidamente identificadas con los nombres y las cédulas de los integrantes del grupo y serán enviadas **por mail** a la dirección de correo electrónico indicada por el docente.
- En la última clase previa a la entrega (viernes 12/07 en Colonia, lunes 15/07 en Montevideo, martes 16/07 en Punta del Este) se tomará una ***defensa oral*** del obligatorio a cada grupo. Consistirá en una serie de preguntas acerca del trabajo, pudiendo abarcar tanto aspectos de diseño como de implementación, y deberá ser realizada por **todos** los integrantes del grupo. El desempeño de cada integrante del grupo en la defensa influirá en la nota final del trabajo.

Objetivo

- Puesta en práctica de diversos conceptos vistos durante el curso.

Planteo del problema

Se trata de una empresa de transporte interdepartamental que quiere informatizar el registro de los recorridos de las líneas de ómnibus que unen distintas ciudades del país. Cada línea realiza una serie de paradas en distintas ciudades a lo largo del recorrido. Por ejemplo, el recorrido de la línea A25 está dado por las siguientes paradas:

- Parada 1: Montevideo
- Parada 2: La Paz
- Parada 3: Las Piedras
- Parada 4: Progreso
- Parada 5: Juanicó
- Parada 6: Canelones

La primera parada de cada línea es la ciudad de origen y la última es la ciudad de destino. Las paradas se van agregando secuencialmente. Cada nueva parada se agrega luego de la que hasta el momento era la última, lo que la convierte automáticamente en el nuevo destino de la línea. Se va a desarrollar un prototipo de un sistema para la gestión de los recorridos de las líneas. En esta primera versión se va a trabajar únicamente con los datos básicos de los recorridos, no interesando contemplar otros aspectos como ser venta de boletos, frecuencia de horarios, choferes, etc.

La cantidad de ciudades que visita la empresa es fija y establecida de antemano (N ciudades). De cada ciudad se registra su número de ciudad y su nombre. Los números de las ciudades irán siendo asignados en forma consecutiva, conforme van siendo registradas en el sistema. La primera ciudad en ser registrada será la número 1, la siguiente será la número 2 y así sucesivamente.

Cada línea de ómnibus se identifica mediante un código alfanumérico que define la directiva de la empresa y junto con el código se registra también la cantidad de paradas del recorrido junto con las paradas correspondientes. De cada parada se registra el número de parada dentro del recorrido y la ciudad correspondiente (número y nombre de ciudad). La primera parada del recorrido tendrá el número 1, la siguiente tendrá el número 2 y así sucesivamente. No existe cota para la cantidad de líneas que posee la empresa ni para la cantidad de paradas que posee una línea (ya que puede haber líneas que pasen más de una vez por una misma ciudad a lo largo de su recorrido).

Las operaciones que se desea realizar en este primer prototipo son las siguientes:

1. Dado el nombre de una ciudad, registrar dicha ciudad en el sistema. Se le asignará en forma automática el número siguiente a la última ciudad registrada hasta el momento. Por ejemplo, si la última ciudad registrada tenía el número 3, la nueva ciudad tendrá el número 4. En caso de que ya se hayan registrado las N ciudades o que ya exista otra ciudad con el mismo nombre, emitir un mensaje de error.
2. Listar número y nombre de cada una de las ciudades registradas hasta el momento en el sistema, ordenadas por número de ciudad de menor a mayor.
3. Dados los números de dos ciudades, registrar un nuevo tramo de recorrido entre ellas. Dicho tramo podrá luego ser utilizado para alguna de las líneas que la empresa maneja. En caso de que dicho tramo ya hubiera sido registrado o que aún no se hayan registrado las N ciudades, emitir un mensaje de error. Se espera que la empresa registre un alto volumen de tramos.
4. Dados los números de dos ciudades, saber si existe alguna secuencia de tramos que las una. Esta operación le servirá a la directiva de la empresa a la hora de idear recorridos para nuevas líneas. En caso de que aún no se hayan registrado las N ciudades, emitir un mensaje de error.
5. Dado el código alfanumérico de una nueva línea, registrar la correspondiente línea en el sistema. Inicialmente, se registrará sin paradas, las que se irán agregando posteriormente. En caso de que ya exista otra línea con el mismo código, emitir un mensaje de error.
6. Listar los datos básicos de todas las líneas registradas hasta el momento (código y cantidad de paradas que posee hasta ahora), ordenadas por código alfanumérico de menor a mayor.
7. Dados el código alfanumérico de una línea y un número de ciudad, agregar una nueva parada en dicha ciudad a su recorrido. Se le asignará en forma automática el número siguiente a la última parada que dicha línea tiene hasta el momento y se sumará 1 a la cantidad de paradas de la línea. En caso de que no exista una línea con ese código, no exista un tramo de recorrido entre la ciudad de la nueva parada y la ciudad de la última parada que dicha línea tiene hasta el momento, o aún no se hayan registrado las N ciudades, emitir un mensaje de error.
8. Dado el código alfanumérico de una línea, listar todas las paradas (número de parada, número de ciudad y nombre de ciudad) de su recorrido, ordenadas por número de parada de menor a mayor. En caso de que no exista una línea con ese código o de que la línea aún no tenga paradas registradas, emitir un mensaje de error.

Se pide:

a) Modelar el problema de las ciudades y los tramos entre ellas mediante un grafo, explicando qué representan vértices y aristas e indicando qué propiedad(es) cumple dicho grafo.

b) Realizar el análisis de Tipos Abstractos de Datos para esta realidad, incluyendo el grafo anterior en el análisis, justificando.

c) Elegir las estructuras de datos más adecuadas para los T.A.D anteriores, justificando la elección y escribir en C++ los tipos de datos correspondientes.

d) Dibujar el esquema de módulos para los T.A.D elegidos, mostrando las inclusiones entre ellos.

e) Implementar el sistema en C++, **respetando** abstracción y modularización. Para interactuar con un T.A.D definido en otro módulo se deben utilizar sus primitivas y otras operaciones necesarias de dicho módulo, **sin** acceder directamente a su estructura de datos.

Observaciones

- Por tratarse de un primer prototipo, en este obligatorio **no** se pide respaldar en archivos la información manejada en memoria. Tampoco se pide eliminar de la memoria la información manejada (se borrará automáticamente cuando el sistema finalice su ejecución).
- En caso de que más de un T.A.D resulte posible para representar un mismo concepto, elegir aquel que resulte más adecuado, justificando apropiadamente.
- En caso de que más de una estructura de datos resulte posible para implementar un mismo T.A.D, elegir aquella que resulte más adecuada, justificando apropiadamente.
- Implemente un módulo aparte para cada T.A.D. En cada módulo, incluya las operaciones primitivas del T.A.D junto con cualquier otra operación adicional que pueda necesitar.
- Al implementar recuerde trabajar **ordenadamente**, partiendo de los módulos más simples y continuando con los más complejos, realizando programas de prueba para cada módulo.
- Los encabezados de las primitivas y otras operaciones incluidas en el documento deben coincidir exactamente con aquellas que luego sean implementadas en los distintos módulos de la aplicación. En particular, el nombre de cada **primitiva** debe coincidir con el nombre provisto para ella en el material teórico del curso.
- Para facilitar la gestión del código fuente y realizar manejo de versiones, se sugiere crear un repositorio en ***GitHub*** durante la realización del trabajo (ver manual correspondiente en la página del curso en moodle).