Normas para la realización del examen: **Duración: 3 horas**

- **El único material permitido durante la realización del examen es un bolígrafo azul o negro.**
- **Debe disponer de un documento oficial que acredite su identidad a disposición del profesor.**
- **No olvide escribir su nombre completo y grupo en todos y cada uno de los folios que entregue.**

Se desea construir la clase `RedMetro` para poder realizar simulaciones sobre una red de transporte. La **red** está compuesta por **líneas**, y cada línea es una sucesión de **paradas**. Ni el número de líneas ni el número de paradas está predeterminado. Algunas paradas pueden ser comunes a distintas líneas. Una parada se identifica por un número (*índice*, toma valores desde 1 y no hay "huecos" en la numeración de las paradas) y puede estar activa -admite el tráfico- para una(s) determinada(s) línea(s) e inactiva (no admite el tráfico) para la(s) otra(s).

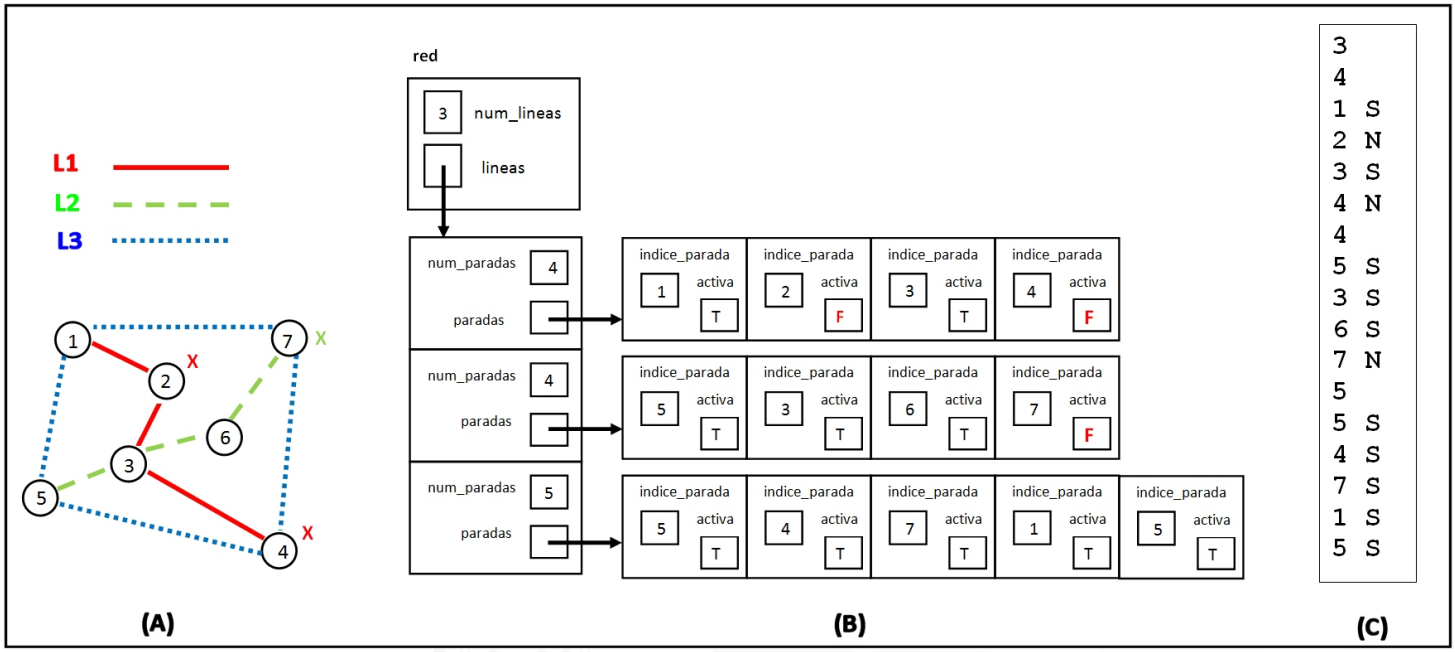
Se propone la siguiente representación para las clases:

```
class RedMetro {
  private:
    int num_lineas;
    Linea * lineas;
  public:
    // ... interfaz
};
```

```
class  Linea {
  private:
    int num_paradas;
    InfoParada * paradas;
  public:
    // ... interfaz
    };
```

```
class  InfoParada {
  private:
    bool activa;
    int indice_parada;
  public:
    // ... interfaz
    };
```

En la figura A mostramos una red formada por tres líneas y siete paradas. Esta red se muestra en la figura B representada en el objeto `red` (clase: `RedMetro`). La numeración de las líneas empieza en 1 y no hay "huecos". La línea 1 es la primera del array referenciado por `lineas`, la línea 2 es la segunda, y así sucesivamente. Las líneas 1 y 2 tienen 4 paradas mientras que la línea 3 tiene 5 (una línea circular *duplica* la parada de cabecera). Las líneas 1 y 2, p.e. comparten la parada 3. En la línea 1 la parada 4 está inactiva, mientras que en la línea 3 esa misma parada está activa.



Para la realización de los ejercicios propuestos deberá escribir `RedMetro.h`, `Linea.h`, e `InfoParada.h`.

Suponga que dispone de los métodos públicos de **consulta** sobre las clases:

- `InfoParada`: a) `GetIndice`, que devuelve el índice de la parada y b) `EstaActiva`, que indica si la parada está activa o no para la línea en la que se encuentra.
- `Linea`: a) `GetNumParadas`, que devuelve el número de paradas y b) operador `[]`, que devuelve una parada(1).
- `RedMetro`: a) `GetNumLineas`, que devuelve el número de líneas, b) `GetNumeroTotalParadas`, que devuelve el número **total** de paradas (en el ejemplo, 7) y c) operador `[]`, que devuelve una línea(1).

(1): El índice de acceso para los operadores [] toma valores 1,2,...

Deberá implementar todo lo que pudiera necesitar y que no se indique explícitamente su disposición.

◁ Ejercicio 1 ▷ Métodos básicos de la clase [1 punto]

Defina los siguientes métodos para la clase `RedMetro`:

1. *(0.5 puntos)* Constructor sin argumentos (crea una *red vacía*) y destructor. Escriba también el método `EstaVacia` que indique si una red está vacía.
2. *(0.5 puntos)* Constructor de copia y operador de asignación.

Nota 1: Los constructores y el destructor de las otras clases están implementados, así como el operador de asignación.

◁ Ejercicio 2 ▷ Sobrecarga de operadores [1.5 puntos]

1. *(0.5 puntos)* Sobrecargue el operador += para la clase `Linea`. El objetivo es incorporar una nueva parada (`InfoParada`) a la línea (`Linea`). El nuevo objeto `InfoParada` se añade **al final** del objeto `Linea`.
   Nota 2: Para soluciones futuras (si fuera preciso) suponga que está implementado el operador += para la clase `RedMetro` (para incorporar una nueva linea (`Linea`) a la red (`RedMetro`).
2. Implemente la siguiente funcionalidad para la clase `RedMetro` sobrecargando los operadores << y >>:
   (a) *(0.25 puntos)* Operador de salida << para poder insertar en un flujo de salida el contenido de una red en formato texto. Deberá aparecer (ver figura C):
   - Un número entero que indica el número de líneas de la red y un salto de línea,
   - Para cada línea de la red:
     - Un número entero que indica el número de paradas de la línea y un salto de línea,
     - Tantas líneas de texto como paradas haya en la línea. En cada una aparecerá: un número entero (el *índice* de la parada), un espacio o tabulador (al menos), un carácter (`S` si está operativa ó `N` si no lo está) y un salto de línea.

   Nota 3: Suponga que está implementado el operador << para las otras clases.

   (b) *(0.75 puntos)* operador de entrada >> para poder obtener desde un flujo de entrada el contenido de un objeto. Se asume el formato indicado en el ejercicio anterior.

◁ Ejercicio 3 ▷ Métodos y funciones para E/S [0.75 puntos]

Implemente un constructor de la clase `RedMetro` con argumento. Recibe el nombre de un fichero de **texto** con la información sobre una red. El fichero debe contener en la primera línea la *cadena mágica* `"METRO"` y un salto de línea. A continuación está el contenido de la red en formato de texto, tal como se indica en el ejercicio 2.

◁ Ejercicio 4 ▷ Sobrecarga de operadores relacionales [0.75 puntos]

Sobrecargue los operadores relacionales ==, != y > para la clase `RedMetro` en base a un valor de calidad. Este valor se calculará a partir de: 1) el número de líneas (30%) y 2) el número de paradas activas (70%).

Nota 4: Una parada común a dos líneas contará dos veces, si es común para tres líneas contará tres veces, ... Si una parada no está operativa para una línea, no contará. Lo hará para cualquier otra línea para la que estuviera operativa.

◁ Ejercicio 5 ▷ Métodos de cálculo [1 punto]

1. *(0.75 puntos)* Implemente el método `MejorConectada` que permita obtener la parada (su *índice*) en la que confluyen el mayor número de líneas, independientemente de si las paradas están activas o no.
2. *(0.25 puntos)* Implemente el método `EstaTotalmenteOperativa` que indique si una línea está totalmente operativa, o sea, si todas sus paradas están activas.

◁ Ejercicio 6 ▷ Aplicación [1 punto]

Escriba un programa *completo* que reciba desde la línea de órdenes el nombre de dos ficheros de texto que contienen la descripción de dos redes de metro (como se indica en el ejercicio 3) y muestre:

1. Cuál es la *mejor* red.
2. Para la mejor red informará sobre la operatividad de cada una de sus líneas.
3. Para la mejor red indicará qué parada es la mejor conectada.