En esta tarea Ud. deberá programar la siguiente función en el archivo *leergrafo.c*:

```
typedef struct nodo {
  char *nom;                  // etiqueta
  int nady;                   // número de nodos adyacentes
  struct nodo **adyacentes;   // arreglo de  nady punteros
} Nodo;
Nodo *leerGrafo(char *nomFile);
```

Esta función lee un grafo almacenado en formato binario en el archivo *nomFile*, entregando el nodo raíz del grafo. El programa *gengrafos.c* escribe 4 grafos en distintos archivos. La función *leerGrafo* debe reconstruir estos grafos. Por lo tanto habrán 4 invocaciones de *leerGrafo* para reconstruir cada uno de estos grafos. Para entender el formato en que está guardado cada grafo, Ud. deberá depurar paso a paso el generador de grafos con el comando: *make gengrafos.ddd*. Lea atentamente los comentarios del programa.

Debido a que no tiene sentido escribir punteros en un archivo, *gengrafos.c* usa un TDA Map (*idMap*) visto en clase auxiliar, para asignar un identificador entero *id* a cada nodo. Dentro del archivo, se usan estos identificadores en lugar de punteros. Dado un nodo, la función *query(idMap, nodo)* permite obtener el identificador del nodo. Durante la lectura del grafo, le será útil usar un mapa invertido, es decir la función *query(idMapInv, id)* entrega el puntero a la estructura de tipo *Nodo* creado para el nodo *id*. Observe que en el mapa los identificadores se disfrazan de punteros, como se explica en esta cátedra en la sección *unboxing vs. boxing*.

Instrucciones

Baje *t4.zip* de U-cursos y descomprímalo. El directorio *T4* contiene los archivos (a) *testgrafos.c* que prueba si su tarea funciona, verificando que reconstruye los mismos grafos escritos por *gengrafos.c*, (b) *grafo.h* que contiene el tipo *Nodo* y el encabezado de la función *leerGrafo*, (c) *pss.c* con la implementación de *Map* y los encabezados en *pss.h*, y (d) *Makefile* que le servirá para compilar y ejecutar su tarea. Se incluye una plantilla en *leergrafo.c.plantilla* con el formato que debe usar para *leergrafo.c*.

Ejecute el comando *make* sin parámetros bajo Debian 11. Le explicará qué requisitos debe cumplir para aprobar su tarea, cuáles son las opciones de compilación y ejecución, cómo entregar su tarea, cómo borrar los archivos intermedios y cuál es el trabajo del comando *make*.

Depuración

Depurar esta tarea puede ser difícil porque si no lee los datos en el orden correcto, la reconstrucción del grafo fallará, sin pista alguna de qué sucedió. Para escribir su función lance *make gengrafos.ddd* para depurar paso a paso la generación de los grafos. Considere especialmente los primeros 3 grafos porque son pequeños para que no lleve mucho tiempo la depuración paso a paso. Fíjese en las instrucciones *fwrite* y los datos que escribe en disco. Piense en cómo leer esos datos en el mismo orden para reconstruir el grafo.

Cómo seguramente cometerá un error, necesitará determinar en qué momento leyó incorrectamente. Le será de mucha ayuda lanzar el comando *make ddd-x2* que lanza 2 *ddd* simultáneamente. El primero es para depurar *gengrafos.c* y el segundo *testgrafos.c*. Depure paso a paso ambos programas y revise que cada vez que escribe un dato con *fwrite* en *gengrafos.c*, en el correspondiente *fread* de la función *leerGrafo*, se lea exactamente el mismo dato. Si son distintos, ahí está el error.

Entrega

Ud. debe entregar por U-cursos el archivo *leergrafo.zip* que genera el comando *make zip*. Recuerde descargar el archivo que subió, descomprimirlo e inspeccionar el contenido para verificar que son los archivos correctos. Se descuenta medio punto por día de atraso, sin considerar recesos, sábados, domingos y festivos. Además se descontará otro medio punto si la compilación arroja *warnings* o no cumple con la indentación requerida en este curso, aunque pase todos los tests.