UTN – FR Mar del Plata – TUP **Laboratorio de Computación II** **Segundo Parcial** Noviembre 2019	Nombre y Apellido:	Nota:

IMPORTANTE:

1. El programa debe compilar sin errores.
2. Agregue comentarios identificando cada inciso realizado.

Una distribuidora de materiales en el país posee una flota de trenes, donde cada tren sólo transporta una determinada categoría de material.
Las **categorías** son:
-Comestible
-Madera
-Metal
-Plástico
-Vidrio

El material llega en **cajones**, cada cajón sólo contiene **una** categoría de material y peso asignado, una vez ingresado al vagón, no es relevante su distinción de los otros.

Condiciones Tren:

- Cada tren posee una cantidad necesaria de vagones, que está determinada por el peso total en cajones que transporta.
- La cantidad **màxima** de peso que puede traccionar cada tren es de **1400kg**.

Condiciones vagón:

- Cuando al ingresar un cajón en el vagón, éste excede su peso, se procederá a ingresarlo en un nuevo vagón.
- Cada vagón posee una carga **màxima** de **200kg**.
- **Inicialmente no existen vagones vacíos**, un vagón se genera sólo si existe un cajón para ingresar.

El archivo **cajones.bin** está contenido por registros del siguiente tipo:

Registro origen de datos
typedef struct { char categoria[20]; float peso; }Cajon;

Se procederá a leer un registro del archivo y distribuirlo a través de una estructura compuesta que considere adecuada y que contenga a las estructuras mencionadas en el siguiente cuadro:

Estructura compuesta		
typedef struct { int nroTren; char categoria[20]; nodo *listaVagones; }tren;	typedef struct { vagon v; struct nodo *sig; }nodo;	typedef struct { int nroTren; float pesoIngresado; }vagon;

Obtenido	Valor	Inciso
	35	1. Diseño y carga exitosa de las estructuras con los datos ingresados a partir de archivo.
	20	2) Realizar las siguientes consultas: a) Informar peso total por **cada categoría.** b) Informar el **porcentaje** de peso cargado sobre **el total** de una determinada categoría. c) Informar tren con **menor** cantidad de peso cargado.
	45	3. Un archivo de pedidos de ciudades posee la siguiente estructura: typedef struct { char ciudad[20]; char categoria[20]; float cantidad; (representada en kg) }Pedido; El archivo està almacenado secuencialmente en orden de pedido. Se pide leer cada registro de archivo, proceder a reducir del vagón la cantidad de material requerida, y si éste se vacía, se procede a **eliminarlo** de la lista correspondiente. Si la cantidad de material requerida **supera** a la almacenada en el tren (la almacenada debe ser mayor a 0), se enviará sólo lo que reste (se enviará menos de lo que pide). Los trenes no se eliminan. Generar un **archivo binario** con el **nro de tren**, contenido **total a transportar** (en kg), y ciudades a las que será transportado.
	-	Hacer un main () que demuestre un correcto funcionamiento de las funciones.

Tabla de puntuación:

Obtenido	10	20	30	40	50	60	70	80	90	100
Nota	1	2	3	4	5	6	7	8	9	10
Condición	**Desaprobado**					**Aprobado**				