Fecha	Curso	Tema	Nota 1p.	Legajo:
				Apellido y nombre:

Antes de comenzar

Forma de trabajo: Complete **V**, **F** o **X**, según considere que la afirmación es correcta, incorrecta, o prefiera omitir responder. Tenga en cuenta que las afirmaciones mal respondidas restan puntaje.

La aprobación del examen requiere un mínimo del 60% en cada uno de los módulos.

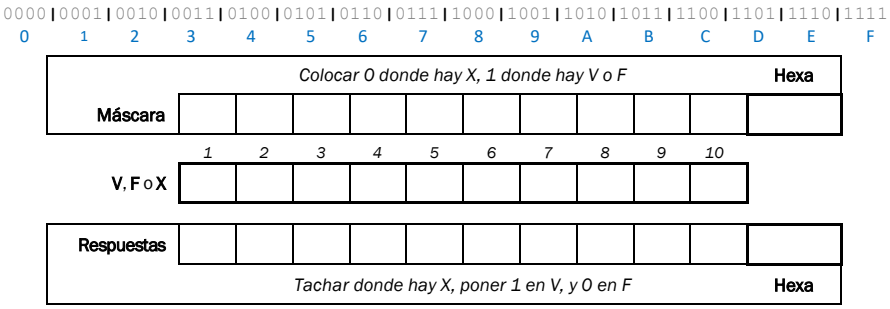
En caso de no haber logrado una calificación numérica en el primer parcial, será condición necesaria aprobar la parte teórica para comenzar con la parte práctica.

Forma de entrega de la parte teórica: Máscara (hexa), Respuestas (hexa) y V, F o X.

Duración total del examen: 2 horas, 30 minutos.

Teórico

1. Llamamos *algoritmo* al modo en que resolvemos un problema.
2. `int*`, `char*` y `short*` permiten declarar punteros. No así `FILE*` que sólo sirve para declarar variables de archivo.
3. Los *arrays* son estructuras de datos indexadas.
4. El acceso directo a cualquiera de los elementos de un *array* conlleva una operación de aritmética de direcciones de memoria.
5. La lista enlazada es una estructura indexada.
6. Un archivo de texto es un archivo que sólo contiene caracteres, no bytes.
7. Las funciones no pueden modificar los valores de sus parámetros.
8. Si una función recibe entre sus parámetros un puntero, entonces sí puede modificarlo, por ejemplo: asignándole `NULL`.
9. Es posible implementar una estructura cola sobre una lista enlazada circular.
10. Para operar con una lista enlazada circular alcanza con tener direccionado el último nodo de la lista.



Práctica

La municipalidad de una ciudad requiere un sistema para administrar y agilizar la gestión de reclamos de los contribuyentes.

Para esto, ponen a nuestra disposición los siguientes archivos:

RECLAMO

```
struct Reclamo
{
   int idRecl;
   int idContrib;
   int idTipoRecl;
   Fecha fechaReclamo;
   char observ[255];
   bool reclamoCerrado;
   Fecha fechaCierre;
   char resolucion[255]
}
```

TIPO_RECLAMO

```
struct TipoReclamo
{
   int idTipoRecl;
   char descrip[50];
}
```

CONTRIBUYENTE

```
struct Contribuyente
{
   int idContrib; // orden
   int dni;
   char nomb[50];
   char dir[255];
   int idTipoContrib;
}
```

El valor de `idTipoContrib` puede ser 1 si es pequeño contribuyente, o 2 en caso de ser gran contribuyente.

Se pide:

1. Por cada tipo de contribuyente, el listado de los reclamos que llevan más de tres meses sin resolución.
2. Porcentaje de reclamos de cada tipo, discriminando por tipo de contribuyente.

Nota: `Fecha` es un TAD. Cualquier función que considere que pueda ser necesaria, simplemente diseñe su prototipo (según las convenciones vistas en clase) y utilícela sin necesidad de implementarla.