```
/**//* 27.940.642-ROSSI, Sebastian Pablo-(05-1965) *//**/
   Ejercicio TDA: Aprobado
   Observaciones: -
**/
/**//* CUALQUIER INCLUDE DE BIBLIOTECA QUE NECESITE, HÁGALO ACÁ *//**//**/
#include <stdlib.h>
/**//* CUALQUIER INCLUDE DE BIBLIOTECA QUE NECESITE, HÁGALO ACÁ
                                                      *//**//**/
/**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//
#include "funciones.h"
/**//* ACÁ DEBE DESARROLLAR LAS FUNCIONES Y PRIMITIVAS PEDIDAS
                                                       *//**//**/
/**//* ADEMÁS DE CUALQUIER OTRA FUNCIÓN QUE SE REQUIERA
                                                       *//**//**/
/**//* FUNCIONES Y PRIMITIVAS A DESARROLLAR
                                                          *//**/
/**//* para la información
                                                          *//**/
void mostrarFinal_MIO(const void *d, FILE *fp)
{
   const tFinal *f = (tFinal *)d;
   if(d)
      fprintf(fp,"%-91d%-*s%-4d%3d\n", f->DNI, (int)sizeof(f->apYNom), f->apYNom, f->codMat, f-
>calif);
   else
      fprintf(fp, "%-9s%-36s%-4s%-4s\n", "D. N. I.", "Apellido(s), Nombre(s)", "Mat.", "Cali");
}
int compararFinal MIO(const void *d1, const void *d2)
{
   return ((const tFinal*)d1)->DNI - ((const tFinal*)d2)->DNI;
}
/**//* para el TDA LISTA
                                                          *//**/
int mostrarLista MIO(const tLista *p,
                void (*mostrar)(const void *, FILE *), FILE *fp)
{
   int cantRegistros = 0;
   mostrar(NULL, fp);
   while(*p)
      mostrar((*p)->info, fp);
      p = &(*p) \rightarrow sig;
      cantRegistros++;
   }
   return cantRegistros;
}
int eliminarYMostrarUnicos MIO(tLista *p, FILE *fpPant,
                        int comparar(const void *, const void *),
```

```
void mostrar(const void *, FILE *))
{
    tLista *q, *inicio = p;
    tNodo *aux;
    int iguales, cont=0;
    mostrar(NULL, fpPant);
    while(*p)
    {
        iguales = 0;
        q = inicio;
        while(*q)
            if(!comparar((*p)->info, (*q)->info))
                iguales++;
            q=&(*q)->sig;
        if(iguales<2)
        {
            mostrar((*p)->info, fpPant);
            aux=*p;
            *p=aux->sig;
            free(aux);
            cont++;
        }
        else
            p=&(*p)->sig;
    }
    return cont;
}
int eliminarYMostrarRepetidos_MIO(tLista *p, FILE *fpPant,
                                   int comparar(const void *, const void *),
                                   void mostrar(const void *, FILE *))
{
    tLista *q, *inicio = p;
    tNodo *aux;
    int f, cont=0;
    mostrar(NULL, fpPant);
    while(*p)
    {
        // la comparacion comienza a partir del siguiente
        // nodo donde esta parado el puntero hasta el final de la lista
        f=0;
        q=&(*p)->sig;
        while(*q)
            // en caso de encontrar un nodo igual
            // muesta, borra y cambia al nodo que es igual
            // para asi seguir comparando
            if(!comparar((*p)->info, (*q)->info))
            {
                f=1;
                mostrar((*p)->info, fpPant);
                aux=*p;
                *p=aux->sig;
                free(aux);
                cont++;
                p=q;
            q=&(*q)->sig;
```

```
}
      // si hay repetidos, al llegar al final de la lista
      // faltara mostrar uno, para eso es la bandera f.
      if(f)
      {
         mostrar((*p)->info, fpPant);
         aux=*p;
         *p=aux->sig;
         free(aux);
         cont++;
         p=inicio;
      }
      else
         p=&(*p)->sig;
   return cont;
}
```