Las sucesivas coordenadas de un móvil se dan como números complejos, que serán almacenadas mediante un programa.

Se usarán las estructuras siguientes:

```
struct HORA {
   unsigned hora,minuto,segundo;
   } ;

struct COMPLEJO {
   float real,imaginaria;
   struct HORA  tiempo;
};

struct lista_COMPLEJOS {
   unsigned total;
   struct COMPLEJO  lista[N];
} ;
```

```
POSICIONES DE UN MOVIL CON COORDENADAS COMPLEJAS
1 Aniadir numero complejo (maximo 8)
2 Eliminar ultimo numero complejo
3 Listar numeros complejos
0 Terminar

posicion de un movil (numero complejo)
  parte real? 1.2
  parte imaginaria? -3.4
hora de la posicion
  hh:mm:ss ? 22:17:54

Numero complejo insertado
PULSE cualquier tecla
```

```
POSICIONES DE UN MOVIL CON COORDENADAS COMPLEJAS
1 Aniadir numero complejo (maximo 8)
2 Eliminar ultimo numero complejo
3 Listar numeros complejos
0 Terminar


complejo: 1.20-3.40i
22:17:54

Numeros complejos listados
PULSE cualquier tecla
```

Figura 1. Ejemplos de ejecución del programa

Se definirán, y **se usarán**, **todas y cada una** de las funciones cuyos prototipos se dan en el siguiente recuadro.

```
//includes
#include "stdio.h"
…

// defines
#define N 8


// prototipos de las funciones que se deben definir en esta práctica

void print_HORA(struct HORA );
void scan_HORA(struct HORA *);
void scan_COMP (struct COMPLEJO *);
void print_COMP(struct COMPLEJO);
void print_menu();
void cualquier_tecla(); // espera que se pulse cualquier tecla, sin hacer echo
void inicializar_lista(struct lista_COMPLEJOS *);
void aniadir_COMPLEJO(struct lista_COMPLEJOS *,struct COMPLEJO);
void eliminar_COMPLEJO(struct lista_COMPLEJOS *);
// elimina el ultimo complejo añadido a la lista
void print_lista_COMPLEJOS(struct lista_COMPLEJOS);


int main(){

    struct lista_COMPLEJOS lis;
    inicializar_lista(&lis);
    char n;
    do{
        print_menu();
        n=getch();
        switch (n){
            case '0':
                break;
            case '1':
                if(lis.total<N)
                    aniadir_COMPLEJO(&lis, lis.lista[lis.total]);
                else
                    printf("No se pueden insertar mas numeros complejos.\n\n");
                cualquier_tecla();
                break;
            case '2':
                if(lis.total>0)
                    eliminar_COMPLEJO(&lis);
                else
                    printf("No hay numeros complejos para borrar.\n\n");
```

```
            cualquier_tecla();
            break;
        case '3':
            print_lista_COMPLEJOS(lis);
            cualquier_tecla();
            break;
        default: printf("Valor incorrecto\n\n");
      }
   }while(n!='0');

return 0;
}
```

// funciones que se deben definir en esta práctica

```
void print_HORA(struct HORA h){
   printf("%.2u:%.2u:%.2u\n\n", h.hora, h.minuto, h.segundo);
}

void scan_HORA(struct HORA *h){
   printf(" - Hora de la posicion (hh:mm:ss)?: ");
   do{
      scanf("%u:%u:%u", &h->hora, &h->minuto, &h->segundo);
   }while(h->hora<0 || h->hora>23 || h->minuto<0 || h->minuto>59 || h->segundo<0 || h->segundo>59);
}

void scan_COMP(struct COMPLEJO *c){
   printf(" - Parte real?: ");
   scanf("%f", &c->real);
   printf(" - Parte imaginaria?: ");
   scanf("%f", &c->imaginaria);
   scan_HORA(&c->tiempo);
}

void print_COMP(struct COMPLEJO c){
   printf("Complejo: %.2f%.2fi\n", c.real, c.imaginaria);
   print_HORA(c.tiempo);
}

void print_menu(){
   printf("POSICIONES DE UN MOVIL CON COORDENADAS COMPLEJAS\n");
   printf("1 A%cadir numero complejo (maximo 8)\n", 164);
   printf("2 Eliminar ultimo numero complejo\n");
   printf("3 Listar numeros complejos\n");
   printf("0 Terminar\n\n");
}
```

```
void cualquier_tecla(){
    printf("PULSE cualquier tecla ");
    getch();
    printf("\n\n");
}

void inicializar_lista(struct lista_COMPLEJOS *inic){
    inic->total=0;
}

void aniadir_COMPLEJO(struct lista_COMPLEJOS *com, struct COMPLEJO c){
    printf("Posicion de un movil (numero  complejo): \n");
    scan_COMP(&c);
    com->lista[com->total]=c;
    com->total++;
    printf("\nNumero complejo insertado.\n\n");
}

void eliminar_COMPLEJO(struct lista_COMPLEJOS *c){
    c->lista[c->total-1]= c->lista[c->total];
    c->total--;
    printf("Numero complejo eliminado.\n\n");
}

void print_lista_COMPLEJOS(struct lista_COMPLEJOS list){
    for(int i=0; i<list.total; i++)
        print_COMP(list.lista[i]);
    printf("Numeros complejos listados.\n\n");
}
```