En la práctica 8, se daban las estructuras siguientes:

```
struct Tfecha {
    int dia, mes, anio;
    };

struct Tcliente {
    char apellidos[40+1];
    struct Tfecha antiguedad;
    float ultimaCompra;
    };

struct TlistaClientes{
        int numClientes;
        struct Tcliente arrayClientes[N];
};
```

para definir una lista de clientes. Utilizando las funciones definidas en la práctica anterior, se deben implementar las funciones descritas en la sección de prototipos, probando su funcionamiento en un programa.

## SOLUCIÓN:

```
// Practica9
// includes y defines
#include <stdio.h>
#include <stdlib.h>

#define N 200


// estructuras

struct Tfecha {
    int dia, mes, anio;
};

struct Tcliente {
    char apellidos[40+1];
    struct Tfecha antiguedad;
    float ultimaCompra;
};

struct TlistaClientes{
    int numClientes;
    struct Tcliente arrayClientes[N];
};


// Prototipos de esta practica
void copiarFechas       (struct Tfecha *, struct Tfecha *);     // copia la segu
nda fecha en la primera fecha
void copiarClientes     (struct Tcliente *, struct Tcliente *); // copia el segu
ndo cliente en el primer cliente
void insertarCliente    (struct TlistaClientes *, int);         // inserta un cl
iente, usando scanCliente, en una posicion que es un integer
```

```
void eliminarCliente     (struct TlistaClientes *, int);         // elimina un cl
iente, de una posicion que es un integer
void vaciarListaClientes (struct TlistaClientes *);          // elimina todos los
 clientes de la lista

// Prototipos de la partica anterior (8)
void scanFecha           (struct Tfecha *);
void printFecha          (struct Tfecha);
void scanCliente         (struct Tcliente *);
void printCliente        (struct Tcliente);
void printListaClientes (struct TlistaClientes);
void inicializarLista    (struct TlistaClientes *);
void anadirCliente       (struct TlistaClientes *);
int longitudLista        (struct TlistaClientes);


// Main
void main()
{
    struct TlistaClientes lst_c;
    inicializarLista(&lst_c);

    printf("\nINSERTAR CLIENTES EN LA LISTA\n");
    do
    {
        anadirCliente(&lst_c);
        fflush(stdin);
        printf("\nanadir otro cliente (s/n)? ");
    } while (getchar() == 's');
    system("cls");

    printf("\nLISTA DE CLIENTES ORIGINAL");
    printListaClientes(lst_c);

    // Ejemplos
    if (lst_c.numClientes > 0)
```

```
    {
        unsigned index;

        printf("\nFECHA COPIADA\n");
        struct Tfecha fecha_ejemplo;
        copiarFechas(&fecha_ejemplo, &lst_c.arrayClientes->antiguedad);
        printFecha(fecha_ejemplo);

        printf("\n\nCLIENTE COPIADO");
        struct Tcliente cliente_ejemplo;
        copiarClientes(&cliente_ejemplo, lst_c.arrayClientes);
        printCliente(cliente_ejemplo);

        printf("\n\nINSERTAR CLIENTE: "); scanf("%u", &index);
        insertarCliente(&lst_c, index);

        printf("\n\nRESULTADO:");
        printListaClientes(lst_c);

        printf("\nELIMINAR CLIENTE: "); scanf("%u", &index);
        eliminarCliente(&lst_c, index);

        printf("\nRESULTADO:");
        printListaClientes(lst_c);

        printf("\nVACIAR LISTA");
        vaciarListaClientes(&lst_c);
        printListaClientes(lst_c);
    }

    printf("\n");
}


// Implementacion de las funciones practica9
void copiarFechas(struct Tfecha * f1, struct Tfecha * f2)
```

```
{
    *f1 = *f2;
}

void copiarClientes(struct Tcliente * c1, struct Tcliente * c2)
{
    *c1 = *c2;
}

void insertarCliente(struct TlistaClientes * lst_c, int index)
{
    if (lst_c->numClientes < N
        && index <= lst_c->numClientes
        && index >= 0)
    {
        int i;
        struct Tcliente new_c;
        scanCliente(&new_c);
        for (i = lst_c->numClientes; i > index; i--)
            lst_c->arrayClientes[i] = lst_c->arrayClientes[i-1];
        lst_c->arrayClientes[i] = new_c;
        lst_c->numClientes++;
    }
}

void eliminarCliente(struct TlistaClientes * lst_c, int index)
{
    if (lst_c->numClientes > 0
        && index < lst_c->numClientes
        && index >= 0)
    {
        int i;
        for (i = index; i < lst_c->numClientes; i++)
            lst_c->arrayClientes[i] = lst_c->arrayClientes[i+1];
        lst_c->numClientes--;
```

```
    }
}

void vaciarListaClientes(struct TlistaClientes * lst_c)
{
    lst_c->numClientes = 0;
}


// Implementación de las funciones practica8
void scanFecha(struct Tfecha * fecha)
{
    do{
        printf("dia? ");
        scanf("%d", &fecha->dia);
    } while(fecha->dia < 0 || fecha->dia > 31);
    do{
        printf("mes? ");
        scanf("%d", &fecha->mes);
    } while (fecha->mes < 1 || fecha->mes > 12);
    do {
        printf("anio? ");
        scanf("%d", &fecha->anio);
    } while (fecha->anio < 0);
}

void printFecha(struct Tfecha fecha)
{
    printf("\n%02d-%02d-%d\n", fecha.dia,
           fecha.mes, fecha.anio);
}

void scanCliente(struct Tcliente * cliente)
{
    printf("\napellidos? ");
    scanf("%s", cliente->apellidos); //Arrays no llevan &
```

```
    scanFecha(&cliente->antiguedad);
    printf("ultima compra? ");
    scanf("%f", &cliente->ultimaCompra);
}

void printCliente(struct Tcliente cliente)
{
    printf("\n\napellido: %s", cliente.apellidos);
    printf("\nantiguedad: %02d-%02d-%d", cliente.antiguedad.dia,
           cliente.antiguedad.mes, cliente.antiguedad.anio);
    printf("\nultima compra: %.2f", cliente.ultimaCompra);
}

void printListaClientes(struct TlistaClientes lst_c)
{
    int i;
    for (i = 0; i < lst_c.numClientes; i++)
        printCliente(lst_c.arrayClientes[i]);
    printf("\n\n");
}

void inicializarLista(struct TlistaClientes * lst_c)
{
    lst_c->numClientes = 0;
}

void anadirCliente (struct TlistaClientes * lst_c)
{
    if (lst_c->numClientes < N)
    {
        lst_c->numClientes++;
        scanCliente(&lst_c->arrayClientes[lst_c->numClientes - 1]);
    }
}
```

```
int longitudLista(struct TlistaClientes lst_c)
{
    return lst_c.numClientes;
}
```