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:

```
// includes y defines

#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

void copiarFechas (struct Tfecha *, struct Tfecha *);
// copia la segunda fecha en la primera fecha

void copiarClientes (struct Tcliente *, struct Tcliente *);
// copia el segundo cliente en el primer cliente

void insertarCliente (struct TlistaClientes *, int);
// inserta un cliente, usando scanCliente, en una posicion que es un integer

void eliminarCliente (struct TlistaClientes *, int);
// elimina un cliente, de una posicion que es un integer

void vaciarListaClientes (struct TlistaClientes *);
// elimina todos los clientes de la lista
```

// main

```
int main(){
     printf("Copia un cliente a otro\n");
     scanCliente(&cliente1);
     printf("\n");
     copiarClientes(&cliente1, &cliente2);
     printf("\n\n");
     printCliente(cliente2);
     printf("Inserta un cliente en una posicion determinada");
     insertarCliente(&lista, 2);
     printListaClientes(lista);
     printf("Elimina un cliente de una posicion\n");
     eliminarCliente(&lista, 2);
     printListaClientes(lista);
     vaciarListaClientes(&lista);

}
```

// implementacion de las funciones

```
void copiarFechas (struct Tfecha * fecha1, struct Tfecha * fecha2){
    fecha2->day = fecha1->day;
    fecha2->month = fecha1->month;
    fecha2->year = fecha1->year;
}


void copiarClientes (struct Tcliente * cliente1, struct Tcliente * cliente2){
    strcpy(cliente2 -> apellidos, cliente1 -> apellidos);
    copiarFechas(&cliente1 -> antiguedad, &cliente2 -> antiguedad);
    cliente2 -> ultimaCompra = cliente1 -> ultimaCompra;
}


void insertarCliente(struct TlistaClientes * lista, int pos){
     int i = 0;
     char apellidos[25 + 1];

     if(lista -> numClientes < N){
          printf("\nEscriba el apellido: ");
          scanf("%s", apellidos);
          for(i = pos; i <= pos && strcmp(apellidos, lista -> arrayClientes[i].apellidos);
i++){
               if(i < lista -> numClientes){
                    scanCliente(&lista -> arrayClientes[i]);
                    lista -> numClientes++;
               }
               else{
               printf("\nEl cliente está en la lista");
```

```
                }
            }
        }
        else{
            printf("\nLa lista está llena");
        }

}


 void eliminarCliente(struct TlistaClientes *lista, int pos){
     int i = 0;
     char apellidos[25 + 1];

     if(lista -> numClientes){
          printf("Escriba el apellido del cliente: \n");
          scanf("%s", apellidos);
          for(i = pos; i < lista -> numClientes && strcmp(apellidos, lista ->
arrayClientes[i].apellidos); i++){
               if(i < lista -> numClientes){
                    lista -> arrayClientes[i] = lista -> arrayClientes[i + 1];
                    lista -> numClientes--;
               }
               else{
                    printf("Cliente no encontrado");
               }
          }
     }
     else{
          printf("La lista esta vacía");
     }

}



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