```c

#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include "lectura.h"
#include "alumn.h"
#include "listaClase.h"

#define MAX_RES 3

/* Definición de prototipos de funciones internas*/

/*****************************************************************************
 Función: haySitioEnLista
    Comprueba si hay sitio en la lista.
    Parámetros de entrada: l, estructura con el array con los datos
        de los alumnos de la clase y el número de elementos de la lista.
    Precondiciones:  la lista l tiene que estar inicializada.
    Parámetro de salida (valor devuelto por la función): booleano, false si no hay sitio,
        true si hay sitio.
 *****************************************************************************/
bool haySitioEnlista (tListaClase l);

/*****************************************************************************
 Función: meterAlumnoEnLista
    Mete en la lista los datos de un nuevo alumno tras el último alumno que había.
    Parámetro de entrada: a, datos del alumno que se quiere meter en la lista de clase.
    Parámetro de entrada/salida: pl, puntero aestructura con el array con los datos de los
        alumnos de la clase y el número de elementos de la lista.
    Precondiciones: tiene que haber sitio en la lista.
 *****************************************************************************/
void meterAlumnoEnLista (tAlumno a, tListaClase *pl);

void inicializarLista (tListaClase *pl){
    pl->elementosLista= 0;
}

int rellenarLista (tListaClase *pl){
    char respuesta[MAX_RES];
    tAlumno alumno;

    printf("Desea meter un alumno en la lista de clase? ");
    leerCadena(respuesta, MAX_RES);
    while (!strcmp(respuesta,"si") && haySitioEnlista(*pl)){
        leerAlumno(&alumno);
        meterAlumnoEnLista(alumno, pl);
        printf("Desea meter un alumno en la lista de clase? ");
        leerCadena(respuesta, MAX_RES);
    }
    if(!haySitioEnlista(*pl))
        printf("\nLo sentimos, la lista de clase esta llena\n");
    return pl->elementosLista;
}

void escribirLista (tListaClase l){
    fprintf(stdout, "Listado de clase\n");
    for (int i=0;i<l.elementosLista;i++){
        printf("\nAlumno %d: ", i+1);
        escribirAlumno(l.lista[i]);
    }
}

bool buscarAlumnoEnLista (const tNombre a, tListaClase l, int *ppos){
    int i;
    bool enc;

```

```
    i= 0;
    enc= false;
    while (!enc && i<l.elementosLista){
        if (!strcmp(a, l.lista[i].nombreAlumno))
            enc= true;
        else
            i++;
    }
    *ppos= i+1;

    return enc;
}

bool modificarAlumnoEnLista(tListaClase *pl, int pos){
    char respuesta[MAX_RES];
    bool modificado;

    printf("Desea modificar los datos del alumno? ");
    leerCadena(respuesta, MAX_RES);
    if (!strcmp(respuesta,"si")){
        modificarAlumno(&(pl->lista[pos-1]));
        modificado= true;
    }
    else
        modificado= false;

    return modificado;
}

bool borrarAlumnoDeLista(tListaClase *pl, int pos){
    char respuesta[MAX_RES];
    bool borrado;

    printf("Desea borrar los datos del alumno? ");
    leerCadena(respuesta, MAX_RES);
    /*Borra colocando el último elemento de la lista en el
    hueco del elemento a borrar*/
    /*if (!strcmp(respuesta,"si")){
        pl->lista[pos-1]= pl->lista[pl->elementosLista-1];
        (pl->elementosLista)--;
        borrado= true;
    }
    else
        borrado= false;*/
    /*Borra desplazando los elementos que están
    a la derecha del elemento a borrar*/
    if (!strcmp(respuesta,"si")){
        for (int i=pos; i<pl->elementosLista; i++){
            pl->lista[i-1]= pl->lista[i];
        }
        (pl->elementosLista)--;
        borrado= true;
    }
    else
        borrado= false;

    return borrado;
}

bool haySitioEnlista (tListaClase l){
    return l.elementosLista<MAX_LISTA;
}

void meterAlumnoEnLista (tAlumno a, tListaClase *pl){
    pl->lista[pl->elementosLista]= a;
    pl->elementosLista++;
```

```
133  }
134
```