Laboratorio 4 C

Main

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "estudiante_v2.h"

/* ------------------------------------------------
            TESTS DE CREAR_LISTA_NOTAS
   ------------------------------------------------ */

void test_crear_lista_notas() {
   struct lista_doble_notas notas;

   printf("test crear lista notas ..... ");
   notas = crear_lista_notas();

   assert(notas.first == NULL);
   assert(notas.last  == NULL);
   assert(notas.size  == 0);
   printf("ok\n");
}

/* ------------------------------------------------
            TESTS DE AÑADIR NOTA
   ------------------------------------------------ */

void test_anadir_una_nota() {
   struct estudiante alumno = {NULL, 0, {NULL, NULL, 0}};

   printf("test anadir una nota ..... ");
   anadir_nota(&alumno, 7.0);

   assert(alumno.notas.first != NULL);
   assert(alumno.notas.last  != NULL);
   assert(alumno.notas.first == alumno.notas.last);
   assert(alumno.notas.first->prev == NULL);
   assert(alumno.notas.last->next  == NULL);
   assert(alumno.notas.size  == 1);

   float value = alumno.notas.first->nota;
   assert(value == 7.0);
   printf("ok\n");
}

void test_anadir_dos_notas() {
   struct estudiante alumno = {NULL, 0, {NULL, NULL, 0}};

   printf("test anadir dos notas ..... ");
   anadir_nota(&alumno, 7.0);
   anadir_nota(&alumno, 6.0);

   assert(alumno.notas.first != NULL);
   assert(alumno.notas.last  != NULL);
   assert(alumno.notas.first != alumno.notas.last);
   assert(alumno.notas.first->prev == NULL);
   assert(alumno.notas.last->next  == NULL);
```

```
    assert(alumno.notas.size  == 2);
    assert(alumno.notas.first->next == alumno.notas.last);
    assert(alumno.notas.last->prev == alumno.notas.first);

    float value1 = alumno.notas.first->nota;
    float value2 = alumno.notas.last->nota;
    assert(value1 == 7.0);
    assert(value2 == 6.0);
    printf("ok\n");
}

void test_anadir_tres_notas() {
    struct estudiante alumno = {NULL, 0, {NULL, NULL, 0}};

    printf("test anadir tres notas..... ");
    anadir_nota(&alumno, 7.0);
    anadir_nota(&alumno, 6.0);
    anadir_nota(&alumno, 8.0);

    assert(alumno.notas.first != NULL);
    assert(alumno.notas.last  != NULL);
    assert(alumno.notas.first != alumno.notas.last);
    assert(alumno.notas.first->prev == NULL);
    assert(alumno.notas.last->next  == NULL);
    assert(alumno.notas.size  == 3);
    assert(alumno.notas.first != alumno.notas.first->next);
    assert(alumno.notas.first->next == alumno.notas.last->prev);
    assert(alumno.notas.first->next->prev == alumno.notas.first);
    assert(alumno.notas.last->prev->next  == alumno.notas.last);

    float value1 = alumno.notas.first->nota;
    float value2 = alumno.notas.first->next->nota;
    float value3 = alumno.notas.last->nota;
    assert(value1 == 7.0);
    assert(value2 == 6.0);
    assert(value3 == 8.0);
    printf("ok\n");
}

/* ------------------------------------------------
        TESTS DE MOSTRAR NOTAS ENTRE VALORES
    ------------------------------------------------ */

void test_mostrar_estudiante_sin_notas() {
    struct estudiante alumno = {"Jose", 19, {NULL, NULL, 0}};

    printf("test mostrar estudiante sin notas..... ");
    mostrar_notas_entre_valores (&alumno, 6.0, 8.0);
    printf("\n");
}

void test_mostrar_estudiante_con_notas() {
    struct estudiante alumno = {"Luis", 18, {NULL, NULL, 0}};

    printf("test mostrar estudiante con notas..... ");
    anadir_nota(&alumno, 7.0);
    anadir_nota(&alumno, 5.0);
    anadir_nota(&alumno, 6.0);
    anadir_nota(&alumno, 9.0);
    anadir_nota(&alumno, 8.0);
```

```
    mostrar_notas_entre_valores (&alumno, 6.0, 8.0);
    printf("\n");
}

/* ------------------------------------------------
            TESTS DE ELIMINAR NOTAS
    ------------------------------------------------ */

void test_eliminar_sin_notas() {
    struct estudiante alumno = {NULL, 0, {NULL, NULL, 0}};

    printf("test eliminar sin notas..... ");
    eliminar_notas(&alumno);

    assert(alumno.notas.first == NULL);
    assert(alumno.notas.last  == NULL);
    assert(alumno.notas.size  == 0);
    printf("ok\n");
}

void test_eliminar_con_notas() {
    struct estudiante alumno = {NULL, 0, {NULL, NULL, 0}};

    anadir_nota(&alumno, 7.0);
    anadir_nota(&alumno, 8.0);
    anadir_nota(&alumno, 9.0);

    printf("test eliminar con notas..... ");
    eliminar_notas(&alumno);

    assert(alumno.notas.first == NULL);
    assert(alumno.notas.last  == NULL);
    assert(alumno.notas.size  == 0);

    printf("ok\n");
}

int main() {
    test_crear_lista_notas();

    test_anadir_una_nota();
    test_anadir_dos_notas();
    test_anadir_tres_notas();

    test_mostrar_estudiante_sin_notas();
    test_mostrar_estudiante_con_notas();

    test_eliminar_sin_notas();
    test_eliminar_con_notas();
}
```

1. Crea el fichero .h de una biblioteca (**estudiante_v2.h**) que contenga las siguientes declaraciones:

```
struct nodo {
   float nota;
   struct nodo *prev, *next;
};

struct lista_doble_notas {
   struct nodo *first;
   struct nodo *last;
   int size;
};

struct estudiante {
   char *nombre;
   int edad;
   struct lista_doble_notas notas;
};

struct lista_doble_notas crear_lista_notas();
/* Crea una lista doble para las notas de un estudiante
   con los campos first y last a NULL, y size a 0.
```

Programa el código del fichero **.c** de esta biblioteca (**estudiante_v2.c**)

2. Añade al fichero **estudiante_v2.h** la declaración de la siguiente función y programa en el fichero **estudiante_v2.c** el cuerpo de la función.

```
void anadir_nota(struct estudiante *p, float nota);
/* Añade un nodo con esta nota al final de la lista de notas
   de un estudiante.
     @param p Puntero a la información del estudiante.
     @param nota Puntuación que se añade a su lista de notas.

   Si el alumno no tiene aún ninguna nota su puntero a la
   lista de notas es NULL.
*/
```

3. Añade al fichero **estudiante_v2.h** la declaración de la siguiente función y programa en el fichero **estudiante_v2.c** el cuerpo de la función.

```
void mostrar_notas_entre_valores(struct estudiante *p,
                                 float desdeNota, float hastaNota);
/* Muestra en pantalla el nombre del estudiante y todas sus notas con
   valores comprendidos entre desdeNota y hastaNota.
     @param p Puntero a la información del estudiante.
     @param desdeNota Puntuación mínima de nota para mostrar en pantalla.
     @param hastaNota Puntuación máxima de nota para mostrar en pantalla.

   Si el alumno no tiene aún ninguna nota su puntero al
   array de notas es NULL.

   El formato de salida a pantalla es el siguiente:
      Nombre: nota1 nota2 nota3 ... notaN
*/
```

4. Añade al fichero **estudiante_v2.h** la declaración de la siguiente función y programa en el fichero **estudiante_v2.c** el cuerpo de la función.

```
void eliminar_notas(struct estudiante *p);
/* Elimina todas las notas de un estudiante.
     @param p Puntero a la información del estudiante.

   Si el alumno no tiene aún ninguna nota su puntero a la lista de notas es NULL.
*/
```

estudiante v2.h

```
#ifndef ESTUDIANTE_V2_H_
#define ESTUDIANTE_V2_H_

struct nodo {
   float nota;
   struct nodo *prev, *next;
};

struct lista_doble_notas {
   struct nodo *first;
   struct nodo *last;
   int size;
};
```

```
struct estudiante {
    char *nombre;
    int edad;
    struct lista_doble_notas notas;
};


struct lista_doble_notas crear_lista_notas();
/* Crea una lista doble para las notas de un estudiante
con los campos first y last a NULL, y size a 0.*/


void anadir_nota(struct estudiante *p, float nota);
/* Añade un nodo con esta nota al final de la lista de notas
de un estudiante.
@param p Puntero a la información del estudiante.
@param nota Puntuación que se añade a su lista de notas.
Si el alumno no tiene aún ninguna nota su puntero a la
lista de notas es NULL.
*/


void mostrar_notas_entre_valores(struct estudiante *p,float desdeNota, float hastaNota);
/* Muestra en pantalla el nombre del estudiante y todas sus notas con
valores comprendidos entre desdeNota y hastaNota.
@param p Puntero a la información del estudiante.
@param desdeNota Puntuación mínima de nota para mostrar en pantalla.
@param hastaNota Puntuación máxima de nota para mostrar en pantalla.
Si el alumno no tiene aún ninguna nota su puntero al
array de notas es NULL.
El formato de salida a pantalla es el siguiente:
Nombre: nota1 nota2 nota3 ... notaN
*/

void eliminar_notas(struct estudiante *p);
/* Elimina todas las notas de un estudiante.
@param p Puntero a la información del estudiante.

Si el alumno no tiene aún ninguna nota su puntero a la lista de notas es NULL.
*/

#endif
```

estudiante v2.c

```
#include "estudiante_v2.h"
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct lista_doble_notas crear_lista_notas(){

    struct lista_doble_notas notas;
    notas.first = NULL;
    notas.last = NULL;
    notas.size = 0;

    return notas;
}
```

```
void anadir_nota(struct estudiante *p, float nota){


    if(p->notas.first == NULL){
        p->notas.first = (struct nodo*)malloc(sizeof(struct nodo));
        p->notas.first->nota = nota;
        p->notas.first->prev = NULL;
        p->notas.first->next = NULL;
        p->notas.last = p->notas.first;
        p->notas.size ++;
    }else if(p->notas.last != NULL ){
        struct nodo nuevo_nodo = (struct nodo)malloc(sizeof(struct nodo));
        nuevo_nodo->nota = nota;
        nuevo_nodo->prev = p->notas.last;
        nuevo_nodo->next = NULL;
        p->notas.last->next = nuevo_nodo;
        p->notas.last = nuevo_nodo;
        p->notas.size ++;
    }

}



void mostrar_notas_entre_valores(struct estudiante *p,float desdeNota, float hastaNota){

    if(p->notas.first == NULL){
            printf("\n%s: ",p->nombre);
        return;
    }

    printf("\n%s: ",p->nombre);

    struct nodo *actual = p->notas.first;
    for(struct nodo *actual = p->notas.first; actual != NULL; actual = actual->next){
        if(actual->nota >= desdeNota && actual->nota <= hastaNota){
            printf("%.2f ",actual->nota);
        }
    }
}


void eliminar_notas(struct estudiante *p){

    for(int i = 0;i < p->notas.size;i++){

        if(p->notas.first == NULL){
            return;

        }else if(p->notas.first->next == NULL){
            free(p->notas.first);
            p->notas.first = NULL;
            p->notas.last = NULL;
            p->notas.size = 0;
            return;
        }else{
            p->notas.first = NULL;
            p->notas.last = NULL;
            p->notas.size = 0;
```

```
            struct nodo *actual = p->notas.first;
            free(actual);
            return;
        }
    }
}
```

Simulacro examen C5

estudiante v3.c

```
#include "estudiante_v3.h"
#include <string.h>
#include <stdlib.h>


struct estudiante nuevo_estudiante_con_N_notas (char *nombre, int edad,
int N, float *notas){

    struct estudiante nuevo_estudiante;

    nuevo_estudiante.nombre = (char*)malloc(MAX_NOMBRE_ESTUDIANTE*sizeof(char));
    strcpy(nuevo_estudiante.nombre,nombre);
    nuevo_estudiante.edad = edad;

    if(notas == NULL) {

        nuevo_estudiante.notas.first = NULL;
        nuevo_estudiante.notas.last = NULL;
        nuevo_estudiante.notas.size = 0;
        return nuevo_estudiante;
    }


        for(int i = 0;i<N;i++){

            if(i==0){

                struct nodo nueva_nota = (struct nodo)malloc(sizeof(struct nodo));
                nueva_nota->nota=*notas;
                nueva_nota->prev = NULL;
                nueva_nota->next = NULL;
                nuevo_estudiante.notas.first = nueva_nota;
                nuevo_estudiante.notas.last = nuevo_estudiante.notas.first;
                nuevo_estudiante.notas.size=1;
                notas++;

            } else {

                struct nodo nueva_nota = (struct nodo)malloc(sizeof(struct nodo));
                nueva_nota->nota=*notas;
                nueva_nota->prev=nuevo_estudiante.notas.last;
                nuevo_estudiante.notas.last->next = nueva_nota;
                nueva_nota->next = NULL;
                nuevo_estudiante.notas.last = nueva_nota;
                nuevo_estudiante.notas.size++;
                notas++;
```

```
            }
            }

        return nuevo_estudiante;


}
```