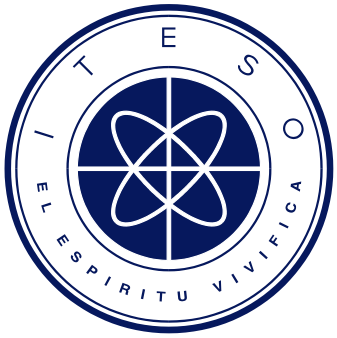
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 1. MANEJO DE APUNTADORES

Autor: DURAN PADILLA, MAURICIO

Presentación: 5 pts

Funcionalidad: 60 pts

Pruebas: 20 pts

24 de mayo de 2018. Tlaquepaque, Jalisco,

Todas las figuras e imagenes deben tener un título y utiilizar una leyenda que incluya número de la imagen ó figura y una descripción de la misma. Adicionalmente, debe de existir una referencia a la imagen en el texto.

La documentación de pruebas implica:
1) Descripción del escenario de cada prueba
2) Ejecución de la prueba
3) Descripción y análisis de resultados.

Instrucciónes para entrega de tarea

Es ***IMPRESCINDIBLE*** apegarse a los formatos de entrada y salida que se proveen en el ejemplo y en las instrucciones.

Esta tarea, como el resto, se entregará de la siguiente manera:

- **Reporte:** vía *moodle* en **un archivo PDF**.
- **Código:** vía su repositorio **Github**.

La evaluación de la tarea comprende:

- 10% para la presentación
- 60% para la funcionalidad
- 30% para las pruebas

Es necesario responder el apartado de conclusiones, pero no se trata de llenarlo con paja. Si no se aprendió nada al hacer la práctica, es preferible escribir eso.

Objetivo de la actividad

El objetivo de la tarea es que el alumno aplique los conocimientos y habilidades adquiridos en el tema de apuntadores para la resolución de problemas utilizando el lenguaje ANSI C.

Descripción del problema

Denisse estudia una ingeniería en una universidad de excelencia, donde constantemente invitan a sus estudiantes a evaluar el desempeño académico de los profesores. Cuando Denisse esta inscribiendo asignaturas para su próximo semestre, descubre que tiene diversas opciones con profesores que no conoce, entonces, decide crear un aplicación que le ayude a ella, y a sus compañeros a seleccionar grupos acorde a los resultados de las evaluaciones de los profesores.

Para iniciar, Denisse solicitó apoyo a traves de Facebook para que sus compañeros de toda la Universidad le apoyaran en la asignación de calificaciones de los profesores. Esto en base a sus experiencias previas en los diversos cursos. La respuesta que obtuvo fue 2 listas de profesores evaluados, la primer lista correspondia a profesores que imparten clases en Ingenierías y la segunda contenia a todos los profesores que imparten clases en el resto de las carreras.

Debido a que Denisse, le gusta programar, decidio crear una pequeña aplicación que le permitiera capturar los datos de los profesores y posteriormente le imprimiera una sola lista con todos los profesores ordenados acorde a su calificación. Lamentablemente, debido a que Denisse salio de viaje, no pudo terminar el programa. Tu tarea es ayudar a Denisse para completar el código.

Código escrito por Denisse

Importante: no modificar el código escrito por Denisse, solamente terminar de escribir el código e implementar las funciones.

```
typedef struct{
        char nombre[15];
        float calificacion;
} Profesor;

float averageArray(Profesor ______  , int _____);
void readArray(Profesor _________, int _____);
void mergeArrays(Profesor _____ , int _____, Profesor ____, int _____, Profesor _____, int ____);
void sortArray(Profesor _____, int ____);
void printArray(Profesor ____ , int _____);


void main(){
        Profesor arr1[20];  //Primer arreglo
        Profesor arr2[20];  //Segundo arreglo
        Profesor arrF[40];  //Arreglo final, con elementos fusionados y ordenados
```

```
        int n1, n2; //Longitud de los arreglos

        readArray(_______________); //leer el primer arreglo

        readArray(_______________); //leer el segundo arreglo

        mergeArrays(_____________________________________);  //Fusionar los dos arreglos en un tercer arreglo

        sortArray(_____________);  //Ordenar los elementos del tercer arreglo, recuerde que pueden
                                   //existir profesores repetidos

        printArray(____________);   //Imprimir el resultado final

        return 0;
}
```

Descripción de la entrada del programa

El usuario ingresara dos listas con máximo 20 elementos (profesores: nombre y calificación). Antes de indicar, uno por uno los datos de los profesores, el usuario debe indicar la cantidad de elementos de la respectiva lista. Así lo primero que introducirá será la cantidad (n1) de elementos de la primer lista (arr1), y en seguida los datos de los profesores de la lista; posteriormente, la cantidad (n2) de elementos de la segunda lista (arr2), seguida por los profesores de los profesores correspondientes.

Ejemplo de entrada:

2
Roberto 7.8
Carlos 8.3

4
Oscar 8.3
Miguel 9.4
Diana 9.5
Oscar 8.5

Descripción de la salida

La salida del programa deberá ser sencillamente la impresión de una lista de profesores y su respectiva calificación (ordenados en orden descendiente, separados por un salto de línea). ¿Qué sucede si tenemos dos o más veces el registro de un profesor? La lista final, deberá mostrar sólo una vez a ese profesor y el promedio de sus calificaciones.

Ejemplo de la salida:

```
Diana       9.5
Miguel      9.4
Oscar       8.4
Carlos      8.3
Roberto     7.8
```

SOLUCIÓN DEL ALUMNO, PRUEBAS Y CONCLUSIONES

Código fuente:

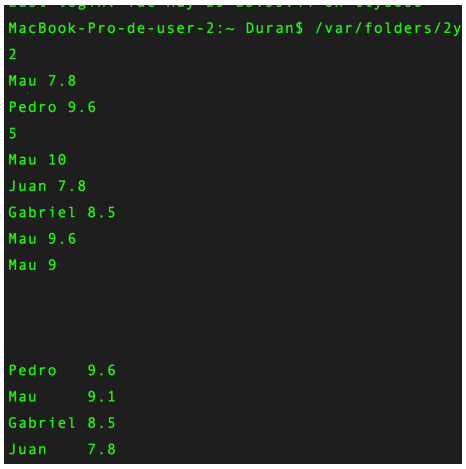
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <ctype.h>

typedef struct{
        char nombre[15];
        float calificacion;
} Profesor;

float averageArray(Profesor * prof, char * nombre, int w); //se modificaron los parametros

void readArray(Profesor * prof, int num);

void mergeArrays(Profesor * prof1, int num, Profesor * prof2, int num2, Profesor * prof3, int num3);

void sortArray(Profesor * prof, int num);

void printArray(Profesor * prof , int num);


int main(void){
        Profesor arr1[20];  //Primer arreglo
        Profesor arr2[20];  //Segundo arreglo
        Profesor arrF[40];  //Arreglo final, con elementos fusionados y ordenados

        int n1, n2; //Longitud de los arreglos

   scanf("%d",&n1);

   readArray(arr1, n1); //leer el primer arreglo

   scanf("%d",&n2);

   readArray(arr2, n2); //leer el segundo arreglo
```

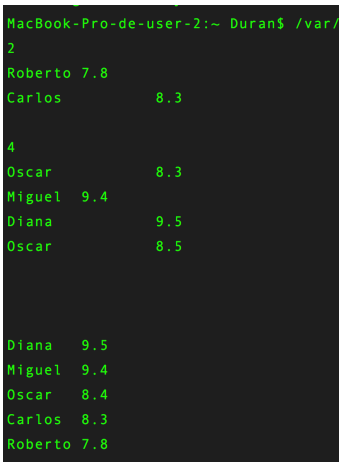
```
    mergeArrays(arr1, n1, arr2, n2, arrF, n1+n2);  //Fusionar los dos arreglos en un tercer arreglo

    sortArray(arrF, n1+n2);  //Ordenar los elementos del tercer arreglo, recuerde que pueden
    //existir profesores repetidos

    printf("\n\n\n");
    printArray(arrF, n1+n2);   //Imprimir el resultado final

    return EXIT_SUCCESS;
}

float averageArray(Profesor * prof , char * nombre, int w){
    float suma = 0;
    float elementos = 0;

    for (int i = 0; i < w; i++) {
        if (strcmp((prof+i)->nombre,nombre)==0) {
            suma+=(prof+i)->calificacion;
            elementos++;
        }
    }

    return suma/elementos;

}

void readArray(Profesor * prof, int num){
    for (int i = 0; i < num; i++) {
        char nombre[15];
        scanf("%s", &nombre);
        strcpy(prof -> nombre, nombre);

        float calif;
        scanf("%f", &calif);

        prof -> calificacion = calif;

        prof ++;
    }
}

void mergeArrays(Profesor * prof1, int num, Profesor * prof2, int num2, Profesor * profF, int num3){
    int condition = 0;
            int w = 0;

    Profesor temporal[40];
    Profesor * temp = temporal;

    for (int i = 0; i < num; i++) {
                        condition = 0;
                        for (int k = 0; k < i; k++) {
            if (strcmp((prof1+i)->nombre,(prof1+k)->nombre)==0) {
             condition = 1;
```

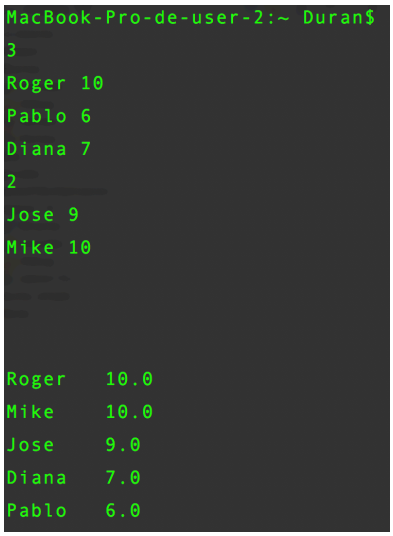
```
            }
        }

        (temp+i)->calificacion = (prof1+i)->calificacion;
        strcpy((temp+i)->nombre,(prof1+i)->nombre);

                    if (condition == 0) {
                        //(profF+w)->calificacion = (prof1+i)->calificacion;
                        strcpy((profF+w)->nombre,(prof1+i)->nombre);
                        w++;
                    }

            }

            for (int i = num; i < num+num2; i++) {
                    condition = 0;
                    for (int k = 0; k < i; k++) {
                        if (strcmp((prof2+i-num)->nombre,(profF+k)->nombre)==0) {
                            condition = 1;
                        }
                    }

        (temp+i)->calificacion = (prof2+i-num)->calificacion;
        strcpy((temp+i)->nombre,(prof2+i-num)->nombre);

                    if (condition == 0) {
                        //(profF+w)->calificacion = (prof2+i-num)->calificacion;
                        strcpy((profF+w)->nombre,(prof2+i-num)->nombre);
                        w++;
                    }
            }

    for (int i = 0; i < w; i++) {
        (profF+i)->calificacion = averageArray(temp,(profF+i)->nombre,num+num2);
    }

}

void sortArray(Profesor * prof, int num){
    Profesor swap;
    for (int i = 0; i < num; i++) {
        for (int k = 0; k < num-1; k++) {
            if ((prof+k)->calificacion<(prof+k+1)->calificacion) {
                swap.calificacion = (prof+k)->calificacion;
                strcpy(swap.nombre, (prof+k)->nombre);

                (prof+k)->calificacion = (prof+k+1)->calificacion;
                strcpy((prof+k)->nombre, (prof+k+1)->nombre);

                (prof+k+1)->calificacion = swap.calificacion;
                strcpy((prof+k+1)->nombre, swap.nombre);

            }
```

```
        }
    }
}

void printArray(Profesor * prof, int num){
    for (int i = 0; i < num; i++) {
                    if (prof->calificacion != 0.0) {
                            printf("%s\t%.1f\n",prof->nombre, prof->calificacion);
                    }
        prof++;
    }
}
```

Ejecución:







Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
- ✓ Lo que me costó trabajo y cómo lo solucioné.
- ✓ Lo que no pude solucionar.

Esta tarea sirvió para repasar lo visto en clase en cuanto a punteros. Puedo decir que aún cuando pude reafirmar lo que sabía de punteros no hubo conocimiento nuevo adquirido.

El principal reto que tuve con esta tarea es que ya se nos daba un prototipo inicial del código y a partir de ahí teníamos que construir las funciones y alguna otra instrucción. Esto fue complicado para mí porque me tuve que apegar a una lógica que no era la mía pero a fin de cuentas las funciones no eran muy complicadas así que sólo requirió un análisis para implementar las funciones.

Otra ligera complicación que tuve fue con la función averageArray ya que aún no sabía cómo checar cuando un nombre se repetía, mi forma de solucionarlo fue agregarle un parámetro (de tipo char*) que nos permite pasar a la función el nombre del que queremos obtener el promedio y compararlo con todos los elementos de un arreglo temporal que contiene el arreglo uno y dos juntos (con elementos repetidos).

Fuera de esos dos percances que se resolvieron con prontitud no hubo mayor problema. Fue una tarea muy interesante.