Arrays unidimensionales, funciones, punteros, struct, ordenamientos y búsquedas

1. Escribe un programa de liquidación de pagos por servicios, para una pequeña compañía que tiene 8 empleados temporales. Con un menú de opciones y para cada empleado/a, el programa recibe los siguientes datos en un array de struct:
    a. el nombre y apellido,
    b. sexo,
    c. horas trabajadas y
    d. sueldo por hora.

Por cada empleado ingresado, el programa deberá mostrar en pantalla en forma ordenada:

- ✓ el nombre
- ✓ el sueldo

Además debe permitir, mediante el menú acceder a la siguiente información:

- ✓ El total de pagos para cada sexo y
- ✓ el sueldo promedio de hombres y de mujeres.
- ✓ Búsqueda binaria de los datos de un empleado mediante el nombre.

El programa debe emitir todos los mensajes necesarios para que el usuario opere en forma eficaz, convertir el apellido de cada empleado a Mayúsculas (ver ejemplo empleo de toupper.c) antes de guardarlo en el array, validar el sexo, permitiendo ingresar minúsculas o mayúsculas indistintamente, horas trabajadas (> a 0 y <=50) y sueldo (base=$250.50 la hora) y controlar al menos 3 errores (ver ejemplo control de errores con cadenas.c). Las búsquedas por nombre y/o apellido deben considerar que el usuario puede ingresar esos datos en minúsculas por lo tanto hay que convertir la cadena antes de realizar el proceso.

2. Escribe un programa, con un menú de opciones, que permita calificar a un grupo de diez alumnos de la escuela secundaria. En un array de struct, se ingresa para cada alumno:
    a. Apellido
    b. Nombre
    c. Campo array de cadena inicialmente vacío
    d. 5 calificaciones de cada alumno, (array de 5 elementos)
    e. Campo float, para el promedio, inicialmente vacío.
    f. Campo float, para la mayor nota, inicialmente vacío.

    El programa debe:

    - ✓ Permitir concatenar nombre y apellido y registrarlo en el campo correspondiente inicialmente vacío.
    - ✓ Permitir calcular la media (promedio) de cada alumno y guardarla en el campo que inicialmente estará vacío hasta que se calcule el dato y se registre.
    - ✓ Permitir buscar la nota mayor de cada alumno y guardarla en el campo que inicialmente estará vacío hasta que se realice el proceso de búsqueda y registro del dato.
    - ✓ Posteriormente debe informar:
        - o el promedio de cada alumno y nota máxima, emitir si está aprobado o no. Para aprobar se requiere un promedio de 6 o más y haber obtenido al menos 6 en la última de las 5 calificaciones.
        - o informar cuántos alumnos aprobaron y cuántos obtuvieron un promedio de al menos 8 puntos.
    - ✓ Permitir la búsqueda (binaria) por Apellido de un alumno y emitir todos sus datos.
    - ✓ Emitir el listado ordenado por nombre y apellido, con sus notas, promedio y nota máxima en forma encolumnada.

    El programa debe emitir todos los mensajes necesarios para que el usuario opere en forma eficaz, validar las notas (>=0 y <=10), convertir el apellido de cada alumno a Mayúsculas (ver ejemplo empleo de toupper.c) antes de guardarlo en el array y controlar al menos 3 errores (ver ejemplo control de errores con cadenas.c). Las búsquedas por nombre y/o apellido deben considerar que el usuario puede ingresar esos datos en minúsculas por lo tanto hay que convertir la cadena antes de realizar el proceso.

3. En un centro médico, se tienen las siguientes especialidades: Alergología, Cardiología, Endocrinología, Geriatría, Pediatría, Nefrología, Neumología, Neurología, Nutriología, Oftalmología, Traumatología. Se registran los turnos el día anterior para el día siguiente y al finalizar el día se deben emitir los listados. Construir un programa con menú de opciones con los siguientes requerimientos:

    a. Cada especialidad puede tener hasta 15 pacientes por día. (campo especialidad y campo array de 15 posiciones enteras inicializadas en 0 (ceros))

    b. Hay 3 médicos por especialidad. Cada médico puede tener hasta 5 pacientes por día. Campos: Nombre, apellido, especialidad y cantidad de pacientes (inicialmente array inicializado en 0 (ceros).

    c. La entidad recibe turnos. Se solicita, nombre y apellido, sexo, dni, fecha de nacimiento. Al pedir turno, cada paciente tiene 2 opciones:

        - elegir por especialidad
        - elegir por nombre y apellido del médico

    d. Se busca disponibilidad por la opción elegida,

        - Si es por especialidad , se ofrecen el listado de médicos con turnos disponibles (alguna posición del arrays en 0), luego se elige entre los médicos con turnos disponibles. Una vez confirmado, se registra con valor 1, la posición libre del array de especialidad y en el array del médico elegido, también se registra en el arrays de pacientes sus datos, especialidad y médico elegido.
        - Si no hay turnos disponibles para la especialidad se debe emitir un mensaje (“Turnos completos, llame mañana”);
        - Si es por médico, se busca si tiene disponibilidad (alguna posición del array libre (en 0)), si la tiene se registra en el arrays con 1 y además en el arrays de la especialidad de ese médico.
        - Si el médico tiene todos los turnos ocupados se debe ofrecer disponibilidad con otros médicos.
        - En el caso de tener los turnos completos, se debe emitir un mensaje (“Turnos completos, llame mañana”);

    e. Al finalizar el proceso se deben emitir los listados correspondientes:

        - Listado de cantidad de pacientes por cada médico.
        - Listado de cantidad de pacientes por especialidad.
        - Listado de pacientes, especialidad y medico.
        - Listado de turnos disponibles por especialidad y médico (para saber si se pueden dar turnos urgentes)
        - Cantidad de turnos otorgados
        - Cantidad de turnos disponibles

    El programa debe emitir todos los mensajes necesarios para que el usuario opere en forma eficaz, convertir el nombre y apellido de cada médico, paciente y especialidad a mayúsculas (ver ejemplo empleo de toupper.c) antes de guardarlo en el array y controlar al menos 3 errores (ver ejemplo control de errores con cadenas.c). Las búsquedas por nombre y apellido deben considerar que el usuario puede ingresar esos datos en minúsculas por lo tanto hay que convertir la cadena antes de realizar el proceso.