Apellido, Nombre: ..... **Mail:** ..... **Padron:** .....

Entrego ..... hojas.				
1	2	3	4	**Nota**

Aclaraciones

- Antes de comenzar a resolver el final, complete sus datos en esta hoja.
- Los ejercicios deben ser implementados en el lenguaje de programación C, respetando las buenas prácticas.
- Para cada ejercicio se recomienda realizar un análisis y un diagrama del problema y la solución.
- Se deben numerar TODAS las hojas e inicializarlas con nombre, apellido y padrón.
- La aprobación del examen está sujeta a la correcta realización de al menos el 60 % del mismo.

Enunciado

La Comarca es una región situada en el noroeste de la Tierra Media. Es habitada por Hobbits, que se caracterizan por su gran hospitalidad.

Últimamente la Comarca fue visitada por personas inescrupulosas que se aprovechaban de la bondad de estos adorables seres, tomándose toda su cerveza, comiéndose toda su comida, yéndose sin lavar los platos y, lo que más les dolió, sin decir gracias.

1 año estuvieron los Hobbits quejándose en silencio, hasta reaccionar al abuso que sufrían.

Merry, quizás un poco ebrio, encabezó la marcha hacia la municipalidad para que el gobernador haga algo.

La solución planteada fue, en principio, registrar quienes entran a La Comarca y puntuarlos, para eso, cerró todas las entradas menos la norte y la sur.

Los visitantes, entonces, se registran al ingresar, actualizando la cantidad de visitas a La Comarca.

Los Hobbits cuentan con 2 archivos, **norte.dat** y **sur.dat**, ambos archivos binarios de acceso secuencial. Cada uno de los archivos está compuesto por registros del tipo **visitante_t** y están ordenados por nombre del visitante.

```c
typedef struct visita {
    char fecha[MAX_FECHA]; // formato AAAAMMDD
    bool trajo_regalos;
} visita_t;

typedef struct visitante {
    char nombre[MAX_NOMBRE];
    visita_t visitas[MAX_VISITAS];
    int cantidad_visitas;
    int id_reputacion;
} visiante_t;
```

Por otro lado tienen un archivo binario de acceso aleatorio, **reputaciones.dat**, de todas las reputaciones posibles, ordenado por **id_reputacion**, los cuales son todos contiguos y empiezan desde el 1, compuesto de registros del tipo **reputacion_t**.

```c
typedef struct reputacion {
    int id_reputacion;
    char descripcion[MAX_DESCRIPCION];
} reputacion_t;
```

Que todo fluya, que nada influya...

C.C.

Ejercicios

1. Teniendo en cuenta que una persona puede entrar desde el norte o desde el sur en distintas visitas, se pide generar el archivo de texto **visitantes.csv**, que unifique los archivos de las entradas (norte.dat y sur.dat) con el siguiente formato:

```
nombre;cantidad_visitas;porcentaje_de_veces_que_trajo_regalo;descripcion_reputacion
```

2. Gandalf tiro unas palabras al aire, al principio creímos que eran insultos porque su club el ComarcaFC perdió el clásico conta el DeportivoHobbit, pero después entendimos que era un hechizo, con el que ordenó el archivo **visitantes.csv** por **descripcion_reputacion**.

Se pide crear un listado que agrupe a los visitantes por reputación, mostrando los nombres de ellos y totalizando según porcentajes de veces que trajeron regalo en sus visitas.

Como se muestra a continuación:

```
Reputación: Alarmante
    Faramir
    Gimli
    ...
    Trajeron regalo:
        Mas del 50% de las veces: 5.
        Menos del 50% de las veces: 64.
------------
Reputación: Buena
    Legolas
    Elrond
    ...
    Trajeron regalo:
        Mas del 50% de las veces: 15.
        Menos del 50% de las veces: 3.
------------
...
```

3. Se volcaron en un vector las reputaciones creadas por los Hobbits. El siguiente algoritmo se supone que busca, recursivamente, una reputación dentro del vector mencionado.

```
int buscar_reputacion(reputacion_t reputaciones[MAX_REPUTACIONES], int tope, int actual, int
    id_buscado){

    if (reputaciones[actual].id_reputacion == id_buscado)
        return actual;

    return buscar_reputacion(reputaciones, tope, actual, id_buscado);
}
```

- **a.** Sin embargo, funciona solo para un caso, ¿cuál?. ¿Cómo lo arreglaría para que funcione correctamente siempre?
- **b.** Muestre con un ejemplo, como funciona el algoritmo corregido.
- **c.** ¿Es la mejor implementación de búsqueda para el vector de reputaciones? Si tiene una mejor, explíquela y justifique por qué es mejor.

4. Teoría

- **a.** ¿Cuál es el motivo por el que utilizamos archivos? ¿Cuáles son las ventajas y las desventajas?
- **b.** ¿Bajo que condiciones se puede volcar el contenido de un archivo a un vector? ¿Qué ventajas y desventajas obtiene?