Ejercicio 1

Dexter y DeeDee miran mucha television estos días, y sus padres, para evitar peleas, decidieron que los días pares quien maneja el control remoto es Dexter y los días impartes DeeDee.
Pero claro, hay varios meses con días impares por lo que DeeDee tendría el control remoto 2 días seguidos (31 y 1).
Ni lerdo ni perezoso, Dexter se percató del problema y le propuso a sus padres que los días 31 se vean programas elegidos por ambos, en particular, aquellos que les gustan a los 2.
Dexter armó un vector con sus programas favoritos y DeeDee hizo lo mismo.

Se pide

1. Crear una estructura que represente un programa visto por Dexter o DeeDee.
2. Dados 2 vectores, uno con los canales elegidos por Dexter y otro con los canales elegidos por DeeDee ambos ordenados por el mismo campo (decidir cual), crear un tercero, que contenga los programas para ver el día 31.

Ejercicio 2

Mojo Jojo cada año elije sus 3 mejores planes malevolos, para usarlos de base en sus planes del próximo año y así ir mejorando. Se sabe que guarda sus planes en un **archivo binario de acceso secuencial**, con la siguiente estructura `plan_t` :

```
typedef struct plan {
    char nombre[MAX_NOMBRE];
    int costo;
    int dificultad; // 1 al 10 inclusives
    int efectividad; // 1 al 10 inclusives
} plan_t;
```

Para poder saber cuales son sus 3 mejores planes primero separa en otro archivo todos los planes a considerar, los cuales deben cumplir con un par de condiciones que le parecen importantes.

- *Aclaraciónes:* el archivo está desordenado, pero se sabe que no hay más de 10 planes con efectividad mayor estricto a 7.

Se pide

1. Dado el nombre del archivo original y el de los planes a considerar, crear este segundo archivo con los planes que tengan un costo menor estricto a 5000 y una efectividad mayor estricta a 7. **Este nuevo archivo debe estar ordenado por efectividad**.

Ejercicio 3

Pardo consiguió trabajo de guardaparque, está muy feliz con su nuevo trabajo porque es lo que ama.
Tiene un par de ideas nuevas para implementar, porque notó que muchas veces la gente se olvida cosas en el camping y quiere poder devolverlas.
Por cada objeto perdido armó un archivo de texto con el siguiente formato:

```
FECHA_HALLAZGO=<aaaammdd>
DESCRIPCION=<descripcion>
```

Además, cuenta con un archivo binario de acceso secuencial de los visitantes, el achivo está ordenado descendentemente por fecha de visita al camping.
Cada registro del archivo es del tipo `persona_t` y contiene la siguente información:

```
typedef struct objeto {
    char descripcion[MAX_DESCRIPCION];
} objeto_t;

typedef struct persona {
    char fecha_en_camping[MAX_FECHA];
    char nombre[MAX_NOMBRE];
    int edad;
    objeto_t objetos[MAX_OBJETOS];
    int cantidad_objetos;
} persona_t;
```

Cabe aclarar, que la fecha en la que se encontró el objeto no necesariamente es la misma en la que la persona estuvo en el camping, pero sí se sabe, que no se pudo haber encontrado algo antes de que el dueño vaya al camping.

Se pide

1. Crear un programa que le permita a Pardo encontrar al dueño de un objeto perdido o quedárselo en caso de que no encuentre al dueño.

Ejercicio 4

Billy y Mandy muchas veces juegan a la batalla naval, por lo general siempre gana Mandy, pero a veces Billy sorprende y gana, y él muy orgulloso de sus victorias, decide guardar el estado de sus barcos en una matriz.

Peeeero, demostrando lo distraído que puede ser, resulta que Billy anotó la matriz al revés. Es decir, puso las filas como columnas y las columnas como filas.

Se pide

1. Crear un procedimiento que dada un matriz y sus respectivos topes, devuelva esta matriz en la forma correcta.

Aclaraciones: - Si la matriz tiene como tope_filas = 3 y tope_cols = 4, deberían quedar como tope_cols = 3 y tope_filas = 4. - El máximo de las filas y columnas es el mismo, es decir, que no habría problema al cambiar los topes.

Ejercicio 5

Billy y Mandy jugaron muchas veces al TaTeTi durante la cuarentena.
Tal es así que se crearon un programita que los ayuda a llevar la cuenta de algo de sus jugadas, pero no nos contaron para qué sirve...
El programa es este:

```
#define MAX_TABLERO 3
#define MAX_JUGADAS 10000
const char CIRCULO = 'O';
const char CRUZ = 'X';

/*
 * Pre: ?????(1)
 * Post: ?????(2)
 */
int gana_ficha(char jugada[MAX_TABLERO][MAX_TABLERO], char ficha) {
    int ganado = 0;
    for (int i = 0; i < MAX_TABLERO; i++){
        if (jugada[i][0] == jugada[i][1] && jugada[i][0] == jugada[i][2] && jugada[i][0] == ficha){
            ganado = 1;
        }
        if (jugada[0][i] == jugada[1][i] && jugada[0][i] == jugada[2][i] && jugada[0][i] == ficha){
            ganado = 1;
        }
    }
    if (jugada[0][0] == jugada[1][1] && jugada[0][0] == jugada[2][2] && jugada[0][0] == ficha){
        ganado = 1;
    }
    if (jugada[0][2] == jugada[1][1] && jugada[0][2] == jugada[2][0] && jugada[0][2] == ficha){
        ganado = 1;
    }
    return ganado;
}

/*
 * Pre: ?????(3)
 * Post: ?????(4)
 */
int resultado_jugada(char jugada[MAX_TABLERO][MAX_TABLERO]) {
    gana_o = gana_ficha(jugada, CIRCULO);
    gana_x = gana_ficha(jugada, CRUZ) * (-1);
    return gana_o + gana_x;
}

/*
 * Pre: ?????(5)
 * Post: ?????(6)
 */
int ?????(7)(char jugadas[100][MAX_TABLERO][MAX_TABLERO], int tope) {
    ?????(8) = 0;
    for (int i = tope-1; i > -1; i++){
        ?????(9) += resultado_jugada(jugadas[i]);
    }
    return ?????(10);
}
```

Se pide

1. Explique **QUE** hace el programa. Para qué lo usan? Qué representa lo que devuelve cada función?
2. Completar los `?????` con las precondiciones, postcondiciones, nombres de funciones y de variables según corresponda.
3. Encuentra alguna mala práctica en el código? En caso de que sí, cuál o cuáles?