Ejercicio 1

Cod.: 101

Las chicas superpoderosas se encargan de mantener la ciudad libre de criminales, es por eso que cada vez que atrapan a uno se encargan de llevarlo a la cárcel. El problema es que se está poniendo cada vez más tedioso organizar las celdas correctamente. Podemos pensar a la cárcel como una matriz de caracteres donde cada posición es una celda y el caracter guardado indica que tipo de criminal hay encerrado.
Sabemos que hay tres tipos de criminales: - Normales: Representados como una N. - Super-Villanos: Representados como una V. - Secuaces: Representados como una S.

Para saber si la cárcel está bien organizada hay que asegurarse que ningún secuaz esté encerrado justo al lado de un super-villano (en cualquiera de las cuatro direcciones).
Este sería un ejemplo de una cárcel bien organizada (los villanos y secuaces no se tocan):

N	V	N	N
N	V	N	S
N	V	N	N
S	N	S	S

Este sería un ejemplo de una cárcel mal organizada (los villanos y secuaces se tocan):

N	V	V	V
N	N	N	S
N	V	N	S
S	S	N	S

Se pide

1. Implementar una función que reciba una matriz de 20x20 y devuelva **true** si la cárcel está bien organizada o **false** en caso contrario.

Ejercicio 2

Cod.: 207

Las chicas superpoderosas están planeando buscar a las mejores heroínas de todo el mundo, porque saben que sus enemigos se están reuniendo y conspirando contra ellas. Una de las heroínas que contactaron para ayudarlas, les pidió un listado de todos los enemigos que conocen hasta ahora, por eso los anotaron en un vector y definieron qué rasgos creen ellas que son importantes:

```
typedef struct enemigo {
    char nombre[MAX_NOMBRE];
    bool tiene_superpoderes;
    int veces_derrotado;
    int veces_ganadas;
} enemigo_t;
```

Se pide

1. Explicar con tus palabras cómo harías para ordenar el vector de enemigos, ascendentemente por nombre. **No está permitido escribir código ni usar ninguna de las palabras reservadas de C**.
2. Una vez que tenian la lista terminada, Burbuja se acordó que hace poco le ganaron una vez más a Mojojojo y no lo habían contado. Crear un procedimiento que busque a Mojojojo y sume uno, a las veces_derrotado.

Ejercicio 3

Cod.: 302

Justo, el humano mala onda de Coraje, está aburridísimo viendo la televisión, en los días cómo hoy, donde nada de lo que ve le gusta, hace lo siguiente, pone el canal 0, espera a que en la pantalla aparezca un número y va a ese canal, luego espera que aparezca un número y va a ese canal, luego espera que en ese canal aparezca un número y va a ese canal, y así sucesivamente hasta que el número que aparece es el 37.

Se pide

1. Crear una función recursiva que reciba un vector de enteros (donde la posición del vector es el canal y el contenido es el número que aparece en pantalla), simule lo que hace Justo y devuelva cuántas veces tuvo que cambiar de canal para llegar al canal que se queda viendo.

Ejemplo:

Número visto	11	8	12	...	76	2	46	...	1	37	9	...
Canal	**0**	**1**	**2**	**...**	**10**	**11**	**12**	**...**	**45**	**46**	**47**	**...**

En este ejemplo, la función debería devolver 4 porque tuvo que ir del canal 0 al canal 11, luego del canal 11 al canal 2, luego del canal 2 al canal 12 y por último del canal 12 al canal 46 para encontrar el 37, es decir, cambió 4 veces de canal.

Ejercicio 4

Cod.: 405

El Señor Conejo tiene ganas de buscar otra mansión, pero necesita que la casa no sea de las más antiguas, porque son las que suelen tener más problemas.

Se pide

1. Teniendo el código del Señor Conejo para reconocer la casa más antigua, enumerar qué buenas prácticas no se cumplen, y explicar qué error de implementación hay y cómo lo solucionarías.

```
#define CINCUENTA 50

typedef struct casa {
    char direccion[CINCUENTA];
    int anio_construccion;
} casa_t;

casa_t casa_mas_antigua (casa_t casas [MAX_CASAS], int t) {

    int anio_mas_antigua = 2000;
    casa_t mas_antigua;

    for (i = 1; i < t; i++) {
        if (casas[i].anio_construccion < anio_mas_antigua) {
            anio_mas_antigua = casas[i].anio_construccion;
            mas_antigua = casas[i];
        }
        i++;
    }
    return mas_antigua;
}
```

Ejercicio 5

Cod.: 510

Los sueños son una experiencia humana universal que puede describirse como un estado de conciencia caracterizado por acontecimientos sensoriales, cognitivos y emocionales durante el sueño. Las pesadillas, son una variante de sueño, donde quien sueña, no la pasa demasiado bien, tal es el caso de Panda, quien en el último tiempo está teniendo demasiadas. Su terapeuta le encomendó que las transcriba, para poder analizarlas. Lo que sí se acuerda, es que todas, tienen un cazador que lo persigue.

Se pide

1. Crear un registro `pesadilla_t` para que Panda pueda transcribirlas, incluir todos los campos que considere que una pesadilla puede tener, por ejemplo lugar, estado del tiempo, nivel de sufrimiento, y como estamos hablando de Panda, un cazador. **Se ruega ser creativo y no limitarse sólo a la información del enunciado**.
2. Dado un vector de pesadillas, determinar qué porcentaje de ellas, son un día nublado y el cazador es Polar.