1.- Serpiente.

Repite el ejercicio de la **Serpiente** que realizamos en el tema de Objetos. La implementación se realizará con listas enlazadas. Si lo hacemos bien el *main* de aquel ejercicio nos debería servir casi en su totalidad.
Cuando esté realizado, generaliza el ejercicio para un **nido** de serpientes que puede albergar, como máximo, hasta 20 serpientes.

2.- Biblioteca.

Repetiremos el ejercicio 9 del tema 6: **Biblioteca**. Al igual que en el ejercicio anterior, si se hace bien, no deberíamos modificar casi nada el *main*.
Define la estructura necesaria que permita manejar los datos de una biblioteca. En dicha biblioteca tendremos libros, DVD's y revistas (tal y como definimos en el ejercicio 3). Diseña el siguiente menú:
1.- Añadir ítem (preguntará que tipo de ítem es y lo añadirá con sus datos a la biblio).
2.- Buscar ítem.
3.- Eliminar ítem.
4.- Listado de la biblioteca
5.- Salir

Realiza el ejercicio de dos maneraas, usando listas y mapas para realizar las operaciones con dichos objetos

3.- Siete y media.

Vamos a realizar el juego de las **siete y media** entre dos jugadores: un humano y la máquina. Al principio se generará un **mazo de cartas** en el que se apilarán 40 cartas y del que se sacan las cartas de los dos jugadores.

Usa una pila para el mazo de cartas.

4.- Solitario.

Como somos informáticos veremos como nuestra vida social se va paulatinamente apagando así que estaría bien ir aprendiendo a jugar a este tipo de juegos.
En el solitario que vamos a programar hay un **mazo de cartas** (baraja francesa) principal que se genera al principio y **dos montones** que, al principio, estarán vacíos. Iremos cogiendo la carta del mazo que está en la cima y la iremos colocando en los montones. Las **reglas** de nuestro solitario serán:
• La base de los montones debe ser un rey de cualquier color.
• Para poner una carta encima de otra en los montones debe ser de diferente color.
• Cuando se acaba de sacar la última carta del mazo principal se debe volver a generar pero respetando el
orden en la que fuimos sacando las cartas.
• Sólo permitimos barajar tres veces.

5.- Round-Robin.

Round-Robin es un algoritmo de planificación de procesos simple de implementar, dentro de un sistema operativo se asigna a cada proceso una porción de tiempo equitativa y ordenada, tratando a todos los procesos con la misma prioridad. En Sistemas operativos, la planificación Round-robin da un tiempo máximo de uso de CPU a cada proceso, pasado el cual es desalojado y retornado al estado de listo, la lista de procesos se planifica por FIFO, del inglés "First In, First Out" (primero en entrar, primero en salir o primero llegado, primero atendido).
Vamos a implementar una **simulación** en la que llegan procesos cada 2 segundos con una identidad (Pnúmero, ejemplo: P1, P2… etc) y un ***quantum*** generado al azar (entre 1 y 4 segundos). Nuestra CPU trabaja durante **0,2 segundos** tras los cuales expulsa al proceso de la CPU y lo mete al final de

la cola. Cuando el proceso agota el tiempo de ejecución sale de la cola. **Debes ir informando de lo que pasa en cada momento.**

Generaliza el ejercicio para procesos con prioridad.

Los procesos tendrán también un **número de prioridad** (0 a 3); a menor número más prioritarios. Cuando se genera un proceso se colocará en su posición en la cola atendiendo a su prioridad. El R-R también respetará esta prioridad.

Ampliación.
Añade una pila en la que el procesador irá **apilando los procesos que vayan terminando**. Cuando la simulación finalice se mostrarán todos los procesos que se ejecutaron sin problemas.

6.- Palíndromo

Usando pilas y colas determina si una cadena que pedimos por teclado es o no palíndroma.