GAME OF THRONES

A partir del relato de Juego de Tronos, desarrollaremos un videojuego sencillo donde el jugador encarna a Qorgyle, un guardián de la noche que lleva semanas caminando al otro lado del muro en una expedición para detectar posibles salvajes. Qorgyle está agotado cuando llega a un poblado abandonado con 12 chozas: allí deberá elegir una y descansar, aunque corre riesgos, pues no todas las chozas están vacías: algunas pueden estar ocupada por un salvaje que haya buscado allí cobijo. Si elige mal, tendrá que luchar por su vida.

1) Diseña estructuras de datos que permitan guardar la información de un guardián, por un lado, y de un salvaje, por otro:
    a. De todo personaje es necesario guardar su nombre, tipo (guardián o salvaje), su nivel de energía y sus reflejos (valor numérico que representa qué porcentaje de los ataques es capaz de esquivar).
    b. Del guardián hay que almacenar además el poder de ataque con sus puños y el poder de ataque con su espada (un valor numérico).
    c. De los salvajes almacenaremos su poder de ataque con lanza (un valor numérico).
2) Diseña ahora estructuras de datos de tipo contenedor para almacenar la información del juego. Concretamente necesitamos dos estructuras más de tipo contenedor:
    a. **Salvajes**: diccionario que almacenará los datos de 5 salvajes.
    b. **Chozas**: una estructura específica guardará información sobre las 12 chozas (número de choza, si está ocupada o no, y personaje que la ocupa).
3) Programa funciones para inicializar –semi aleatoriamente– las estructuras creadas:
    a. El nombre de cada personaje se solicita al usuario.
    b. Los reflejos del guardián oscilan entre el 40 y el 60% (simúlalo mediante un valor aleatorio) y los de los salvajes, cada uno de ellos distinto, pero siempre entre 25 y 40% (de nuevo, simúlalo).
    c. Para los datos niveles de energía ten en cuenta que va siempre de 0 a 100. Inicialicemos la energía de Qorgyle a 50 pues está agotado y la de cada salvaje a un valor aleatorio entre 70 y 100 (están descansando).

d. En cuanto a los reflejos, depende: los de Qorgyle son entre el 20 y el 60% (simúlalo mediante un valor aleatorio, pero menos si está cansado, como es el caso) y los de los salvajes es distinto para cada uno, pero siempre entre 25 y 40% (de nuevo, simúlalo).

e. Para los valores de ataque: Del guardián hay que almacenar el poder de ataque con sus puños (20) y con su espada (45). De los salvajes almacenaremos su poder de ataque con lanza (un valor entre 30 y 40 que debe simularse aleatoriamente).

f. Para las chozas: como esta inicialización es un proceso previo a que el guardián intente acceder al poblado, de estar ocupada una choza solamente podrá estarlo por un salvaje y no por el guardián. Así, para cada una deberá saberse si está vacía u ocupada (las posibilidades de que haya un enemigo son del 15% y de que esté vacía del 85%). Si está ocupada, se buscará un salvaje y se enlazará con él.

IMPORTANTE: "chozas" y "personajes" deben ser 2 estructuras separadas, pero cuando a una choza entra un salvaje (en la inicialización) deben enlazase el salvaje que hay en "personajes" y el que está en "chozas". Deben ser el mismo enlazado, p.ej.:

Choza3 → salvaje 2 ← Personajes

4) Justo antes de dormir, el guardián consulta un oráculo para saber el nivel de riesgo de elegir choza. El oráculo le permite saber cuántas chozas hay libres sin abrir ninguna puerta. Escribe un subprograma en Python que, a partir de las chozas creadas en el apartado (2), calcule recursivamente cuántas hay ocupadas o libres siguiendo estas ideas:

   a. La función oráculo recibe un parámetro que indica si se consulta el número de cabañas libres u ocupadas.

   b. El oráculo mira la primera posición, computa si está libre u ocupada y llama recursivamente a la función para el resto de las chozas.

   c. Finalizado el proceso se retorna el número computado (libres u ocupadas)

5) Con los resultados del oráculo, el guardián de la noche debe elegir una choza para dormir. La elección se simulará con un proceso aleatorio cuyo resultado tiene importantes consecuencias en el juego: si la choza elegida estaba vacía, el guardián se va a dormir, en caso contrario ha de luchar con el enemigo. Dormir le aporta un nivel 100 de energía y beneficia sus reflejos: su poder para esquivar ataques sube al 50%. La lucha se simula en otro apartado, pero mientras, programa en Python un subprograma que simule esta sencilla parte del juego.

6) Elabora un subprograma en Python que encuentre cuál es el enemigo más temible de todos los que se encuentran emboscados en alguna choza y cuál el menos. Elabora tú, de forma creativa, un método para determinar cuándo un enemigo es más o menos temible.

7) Ordena la estructura de chozas de la siguiente manera: primero, todas las cabañas vacías y después las que están ocupadas. Dentro de las ocupadas, primero las ocupadas por enemigos y después la ocupada por el guardián, y dentro de las ocupadas por enemigos, por orden alfabético de nombre.

8) Cuando el guardián elige una choza ocupada por un salvaje, se desencadena una lucha por la supervivencia que consiste en un proceso donde iterativamente el guardián y su enemigo van intercambiando golpes hasta que uno pierde toda su energía y muere. El guardián parte con menos energía —caminar en la noche ha mermado sus facultades— pero cuenta con el factor sorpresa y ataca primero. El guardián elige aleatoriamente cómo atacar: puños o espada; si bien tiene una destreza sobresaliente con los puños (cada golpe merma 20 de vida a su oponente), cuando usa su espada es cuando sus golpes son verdaderamente terribles (merma 40 de vida a su oponente). Usa puños el 30% de las ocasiones y espada en el restante 70%. En cuanto a la defensa, está cansado, así que solo es capaz de esquivar el 10% de los ataques de sus enemigos (inicializa a este valor antes de iniciar la pelea). El salvaje, que parte de un nivel de energía alto por haber ya dormido parcialmente, solo cuenta con su lanza con la que ataca el 100% de las ocasiones. Programa un proceso donde alternativamente cada uno de los luchadores intente un ataque y el otro elija una respuesta, tras lo cual se actualicen los niveles de vida de ambos hasta. Este proceso debe repetirse hasta que la energía de uno de los dos quede a cero y muera. En ese momento el juego anunciará si el guardián pudo finalmente descansar o si por el contrario murió en el intento de ocupar la cueva.