Universidad Nacional de La Matanza

Algoritmos y estructuras de datos

Trabajo práctico:

Cmon dice

Necesidad

El departamento de recursos humanos de CodeInc, una startup de sistemas, está planeando una actividad recreativa. Se les ocurrió hacer un torneo en donde cada participante tenga que jugar al “Cmon dice” (Simon dice) y el que más puntos obtenga recibirá un premio.

Reglas del juego

Al inicio de cada juego, se va a:

- Ingresar los nombres de los jugadores
- Elegir la dificultad (fácil, medio, difícil)
- Sortear el orden de los jugadores
- Mostrar las configuraciones del juego (cantidad de vidas para cada participante, cantidad de segundos en los que tienen que ingresar la secuencia, cantidad de tiempo que van a poder ver la secuencia)

Luego:

- Empieza el primer jugador sorteado
- El sistema le muestra la primera secuencia al jugador durante X tiempo, una letra a la vez.
- El jugador tiene Y segundos para ingresar la secuencia, letra por letra. Cualquier otro caracter ingresado debe ser ignorado a menos que sea el de usar las vidas
- Si se le acaba el tiempo y no contesta, automáticamente se indicará que el tiempo finalizó. Si tiene una vida, puede volver para atrás N cantidad de jugadas (según vidas tenga) y tiene un segundo intento. Si no tiene vidas, el juego finaliza para esa persona y le toca al siguiente participante.
- Si ingresa correctamente la secuencia, el programa le muestra al jugador la siguiente secuencia
- Si ingresa la secuencia, pero es incorrecta, el programa le va a indicar que puede deshacer los últimos movimientos, según la cantidad de vidas que tenga.
- Por cada round se van sumando puntos, y el jugador que obtenga la mayor cantidad de puntos es el ganador.

Los puntos de cada jugador de la siguiente manera:

- Por cada secuencia bien ingresada, sin usar vida, se recibe **+3 puntos**
- Por cada secuencia bien ingresada, pero usando vida, se recibe **+1 punto**

Ganará la persona con mayor cantidad de puntos. En caso de que hayan 2 personas con la mayor cantidad de puntos (ejemplo: Jugador 1 y jugador 3 hicieron 10 puntos cada uno, mientras que el jugador 2 hizo 7 puntos), entonces el jugador 1 y 3 serán los ganadores.

Nota 1: Los niveles de dificultad son:

- Fácil
- Medio
- Difícil

Estos niveles de dificultad tienen que ser configurables mediante un archivo txt.

Ejemplo:

Dificultad elegida: Media. Tiempo que se muestra la secuencia por pantalla: 5 segundos. Cantidad de tiempo que tiene la persona para contestar: 10 segundos. Cantidad de vidas: 1

Jugador 1		**Jugador 2**		**Jugador 3**	
Secuencia	***Respuesta***	***Secuencia***	***Respuesta***	***Secuencia***	***Respuesta***
A	A **(+3 puntos)**	**V**	V **(+3 puntos)**	**N**	N **(+3 puntos)**
AV	AV (uso una vida) **(+1 punto)**	**VR**	VR **(+3 puntos)**	**NA**	NA **(+3 puntos)**
AVR	AVR **(+3 puntos)**	**VRV**	No contesta. Usa una vida para volver a ver la secuencia. VRV **(+1 punto)**	**NAA**	NAA **(+3 puntos)**
AVRR	AVRV **(0 puntos)**	**VRVA**	VRVA **(+3 puntos)**	**NAAV**	NAAR (usa una vida, vuelve para atrás y contesta mal de nuevo) **(0 puntos)**
-		**VRVAR**	VRVAA **(0 puntos)**	-	
7 puntos		**10 puntos**		**9 puntos**	

Al final de cada juego se deberá generar un informe que indique las secuencias de cada round, qué contestó cada jugador, cuántas vidas usaron en cada round y cuál fue el puntaje obtenido por pregunta, puntaje total y quién/es ganaron el premio.

Consigna

Hacer un programa en C para jugar este juego.

Las configuraciones iniciales serán leídas de un archivo txt (config.txt) con el siguiente formato:

```
Nivel | Cant. de tiempo que se muestra las secuencia completa | Cant. de tiempo que tiene el jugador para contestar | Cant. de vidas
```

```
F | 10 | 20 | 3
M | 8 | 15 | 2
D | 5 | 10 | 0
```

La cantidad de vidas no puede ser superior a 5, y los tiempos para jugar no pueden superar los 20 segundos, ya sea para mostrar por pantalla o el tiempo que se le da a la persona para responder.

Si el archivo no tiene un formato válido, el programa lo debe indicar.

Apenas se inicie el programa, deberá haber un menú de 2 opciones:

[A] Jugar

[B] Salir

Si alguien ingresa a “Jugar”, primero se le pedirá que cargue los nombres de las personas que van a jugar. Puede ingresar la cantidad de nombres que desee.

Una vez que termine de ingresarlos, tiene que elegir el nivel de dificultad del juego (fácil, medio, difícil) y entonces aparecerá por pantalla el orden en el que jugarán los jugadores (recordar que se elige aleatoriamente), los tiempos/configuraciones de la partida, las teclas que puede tocar y se le preguntará al primer jugador si está listo. En caso de que sí, inicia el juego.

Por round, se le muestra una secuencia al jugador, letra por letra, y se espera la cantidad de segundos establecida en el archivo de configuración para luego eliminar la secuencia de la pantalla y dejar que el jugador conteste. Tiene la cantidad de segundos establecida para ingresar todas las letras de la secuencia.

Al terminar de ingresar la secuencia, el programa debe validar si la misma es correcta o no. En caso de que no sea correcta, y tiene vidas disponibles, se le muestra por pantalla que se equivocó y se le muestra la cantidad de vidas que tiene disponible. En ese momento, tiene que ingresar cuantas vidas (jugadas hacia atrás) desea deshacer, y se le vuelve a reiniciar el tiempo para contestar dicha secuencia.

Al finalizar el turno de todos los jugadores, el sistema mostrará por pantalla quién ganó y por cuantos puntos, y a su vez va a generar un informe de que fue lo que sucedió en esa partida, indicando las secuencias de cada round, qué contestó cada jugador, cuál fue el puntaje obtenido por pregunta, cuántas vidas se usaron en cada round, puntaje total, y quién/es ganaron el premio. El nombre del archivo debe contener la fecha y la hora actual en el siguiente formato: YYYY-MM-DD-HH-mm. Ejemplo de nombre: informe-juego_2024-02-01-12-20.txt

Nota: La secuencia va a estar formada por estos 4 colores (R = rojo, V = verde, A = amarillo, N = naranja). Pueden mostrar colores o letras a la persona que esté jugando, lo dejamos a la creatividad de cada grupo.

Además, deberán subir el código a un repositorio en GitHub. Deberán agregar a su repo un readme que indique cómo jugar el juego y qué hacer si quiero cambiar las configuraciones del juego.

A ese repo se deberá subir un documento con diferentes lotes de prueba con el siguiente formato:

Descripción	Salida esperada	Salida obtenida
Se quiere probar qué es lo que pasaría si….	Se espera que….	La salida obtenida fue….

Mínimo se deben documentar 8 casos de prueba, con captura de pantalla de la salida obtenida.

Vidas

Las vidas se pueden usar en 3 situaciones:

1. Si no se ingresó ninguna letra de la secuencia y el tiempo finalizó:
    a. En este caso solo se puede volver 1 vida para atrás y que el programa muestre la secuencia nuevamente
2. Si se ingresaron todos los caracteres de la secuencia, y hay alguno equivocado
    a. En este caso, se pueden volver para atrás N cantidad de caracteres, según la cantidad de vidas que tenga la persona.
    b. Ejemplo:

        Secuencia mostrada por pantalla **R A V N N**

        Secuencia ingresada por el jugador: **R A N N N**

        Cantidad de vidas disponibles: 2

        Jugador selecciona que quiere usar sus 2 vidas

        Entonces por pantalla se le muestra **R A N** y tiene que completar a partir de ahi los colores restantes. En este caso, perdería de cualquier manera ya que el error estuvo en la 3era letra ingresada, y solo podía volver para atrás 2 letras.
3. En el medio del ingreso de la secuencia
    a. En este caso puede apretar una tecla (a definir por el grupo) y volver N cantidad de jugadas hacia atrás, según vidas tenga. Se puede ingresar hasta una vida más de la cantidad de letras que se escribió, y en caso de que esto suceda se muestra nuevamente la secuencia
    b. Ejemplo

        Secuencia mostrada por pantalla **R A V N N**

        Secuencia ingresada por el jugador: **R**

        Cantidad de vidas disponibles: 2

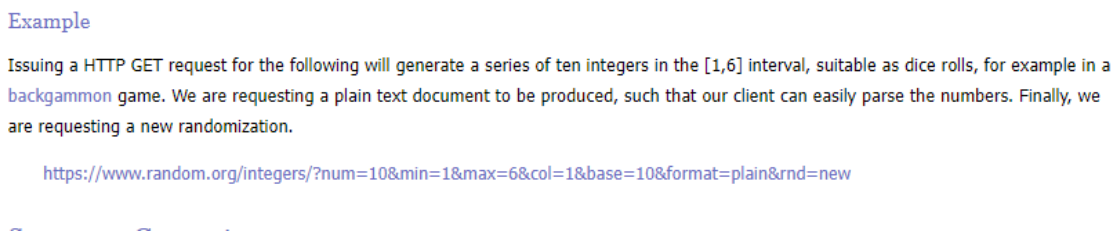
El jugador toca que quiere usar sus vidas y vuelve 2 jugadas hacia atrás. Entonces deshace la R y se le muestra nuevamente por pantalla la secuencia completa. En caso de que seleccionase de que quiera usar 1 vida, solamente se le borra la R y se le habilita el ingreso de la secuencia sin mostrarle por pantalla la secuencia original.

Secuencia

Para determinar la secuencia de cada juego (A, R, V, N), se va a consultar a una API externa que genera números aleatorios. En base a los números generados, se lo va a asociar a un color y esa será la secuencia para cada jugador. Cada jugador juega una secuencia “única”.

Para esto, van a usar la API que se encuentra documentada en la siguiente página:

https://www.random.org/clients/http/api/



Example

Issuing a HTTP GET request for the following will generate a series of ten integers in the [1,6] interval, suitable as dice rolls, for example in a backgammon game. We are requesting a plain text document to be produced, such that our client can easily parse the numbers. Finally, we are requesting a new randomization.

https://www.random.org/integers/?num=10&min=1&max=6&col=1&base=10&format=plain&rnd=new

Entonces, desde el código hacen una llamada HTTP a esa URL para cada uno de los jugadores, para traerse random los números. Tienen que asociar a cada número un color y esa va a ser la secuencia para ese jugador.

Condiciones básicas para aprobar

- ✔ Código prolijo, dividido en funciones
- ✔ Funciones lo más genéricas posibles
- ✔ Nombres significativos de variables
- ✔ 0 errores 0 warnings
- ✔ Que funcione mínimo para todos los casos de prueba que presentan