Juego de memoria

Fecha de entrega: 28/agosto.

Introducción

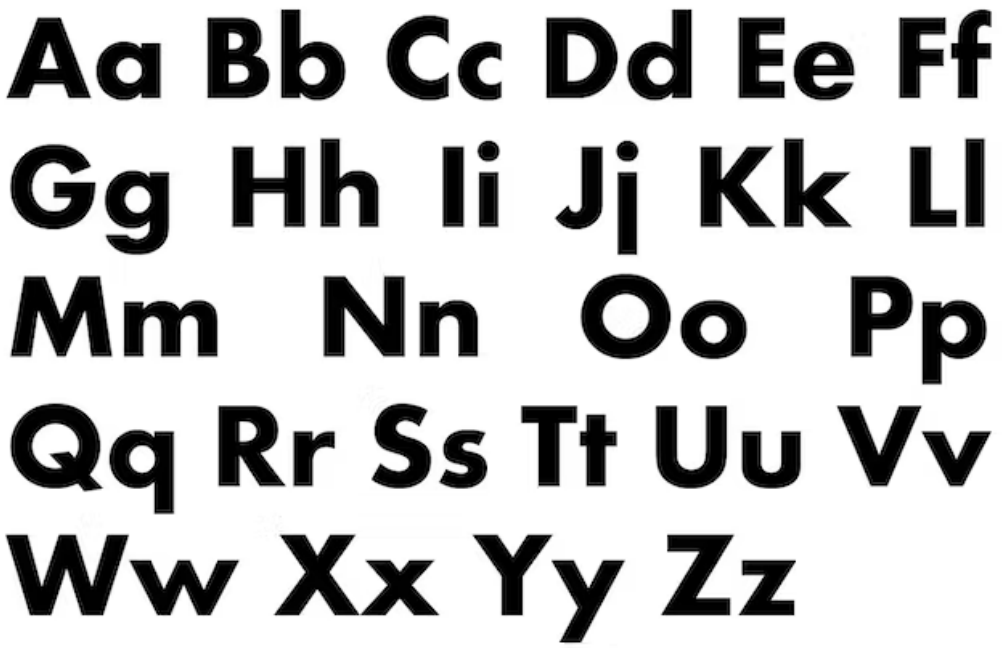
Para esta actividad vas a completar este juego de memoria de letras, el cual consiste en encontrar todos los pares en 6 turnos o menos.

Tendrás a disposición el juego avanzado en un 50% y te corresponde terminar el 50% del mismo.

Son dos archivos con los cuales se debe trabajar: memory_utils.ex y memory_tasks.ex.

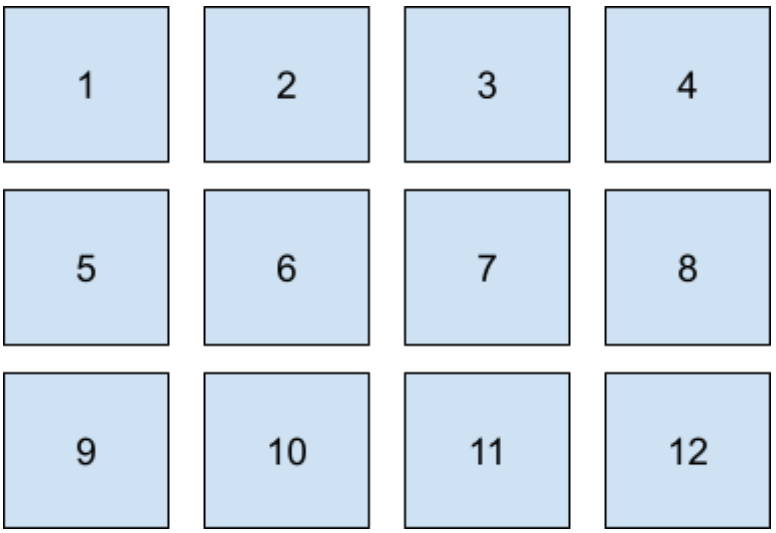
Contexto

El juego simula un juego de memoria con las 26 letras del alfabeto.



Al usuario se le muestra una especie de tablero, con 12 “cartas”. Esto implica que de las 26 letras del alfabeto, se escogen6 aleatoriamente. Además, estas deben repartirse de la siguiente manera: 3 vocales y 3 consonantes.

El tablero debe lucir aproximadamente de la siguiente manera:



La numeración sí debe ser tal y como se muestra, además de contar con esta distribución: 3 filas y 4 columnas.

El jugador debe hallar los 6 pares para ganar, que consisten en la misma letra en mayúscula y en minúscula. Las coordenadas son bidireccionales, es decir que por ejemplo (2,4) y (4,2) podrías apuntar a un mismo par correcto o incorrecto.

Ejemplo de pares correctos: (A,a) ; (z, Z) ; (M, m) ; (j, J)
Ejemplo de pares incorrectos: (C,N) ; (g, H) ; (e, u) ; (S, k)

Debe llevar la cuenta de los pares que va encontrando, separando entre vocales y consonantes. Las rondas son ilimitadas pero, tendrá 3 oportunidades/vidas, para equivocarse.

El juego termina cuando se acaban las oportunidades o cuando se encuentran todos los pares.

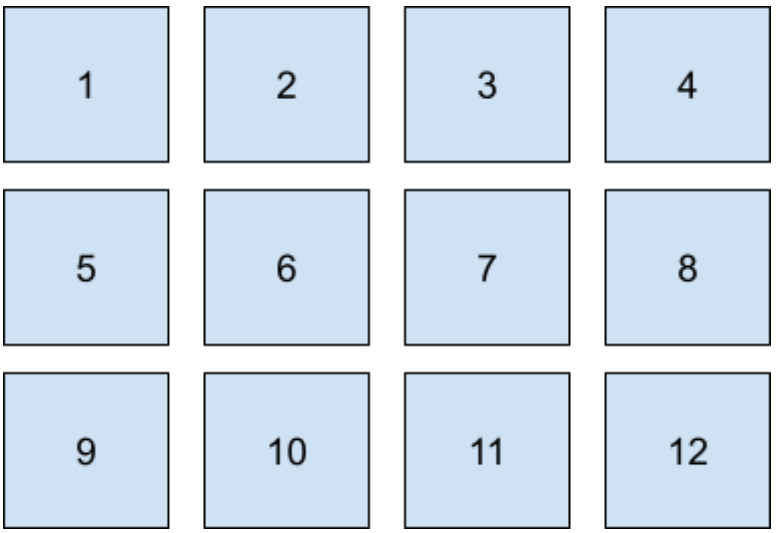
Ejemplo de ejecución

Juego de memoria - Elixir + PF

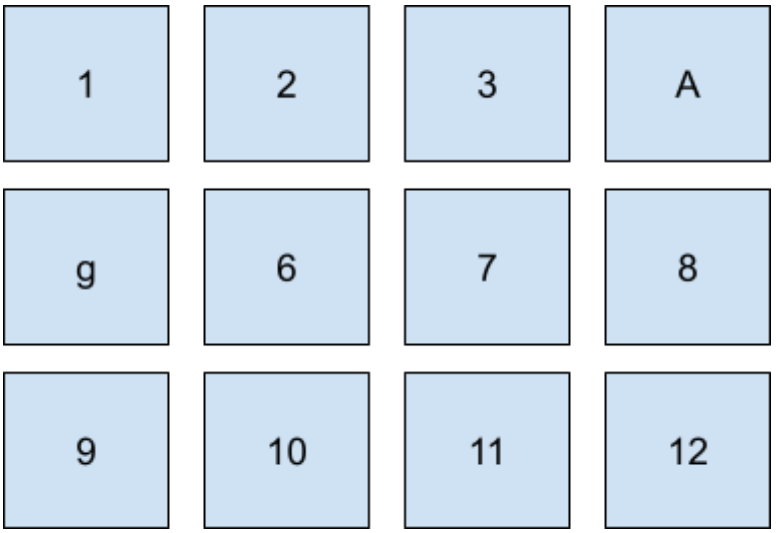
Ingrese nickname: mdyagual

Jugador : mdyagual

Pares vocales: 0
Pares consonantes: 0
Vidas: 3

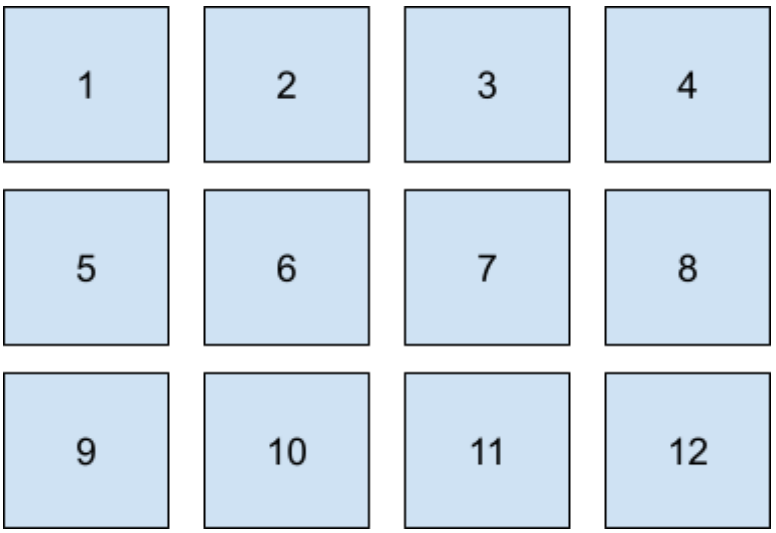


Seleccione el par: 4,5

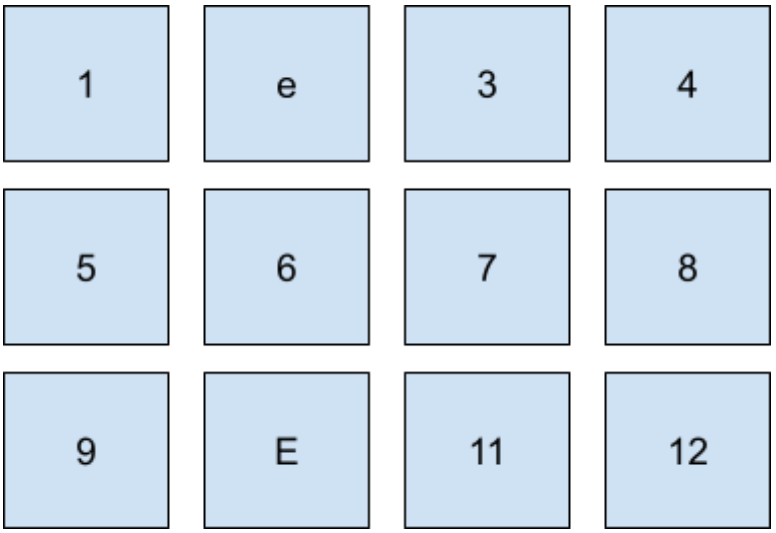


Sigue intentando.

Jugador : mdyagual
Pares vocales: 0
Pares consonantes: 0
Vidas: 2

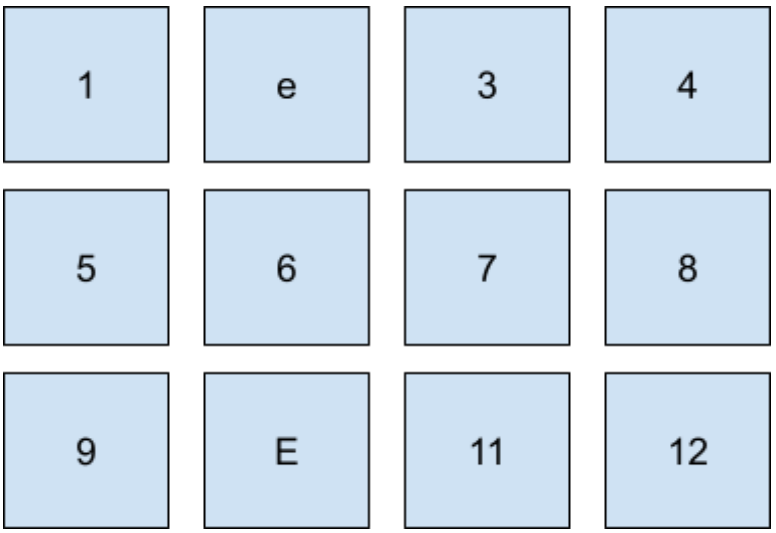


Seleccione el par: 2,10

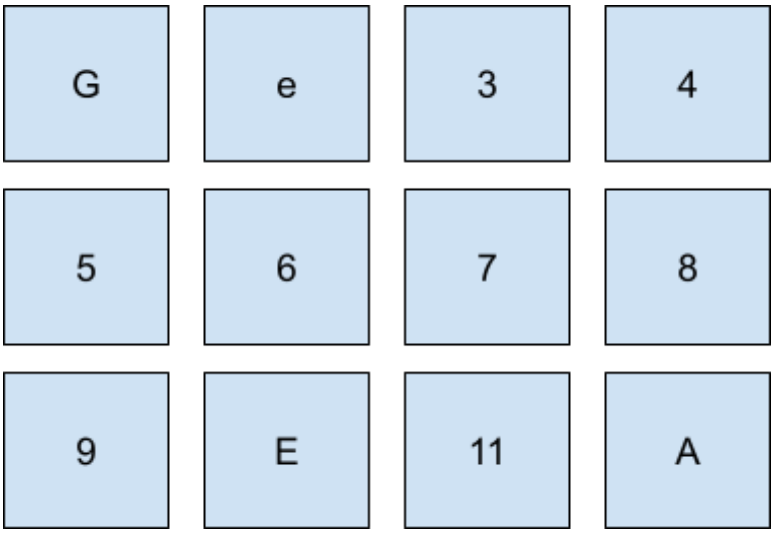


Has encontrado un par de vocales.

Jugador : mdyagual
Pares vocales: 1
Pares consonantes: 0
Vidas: 2

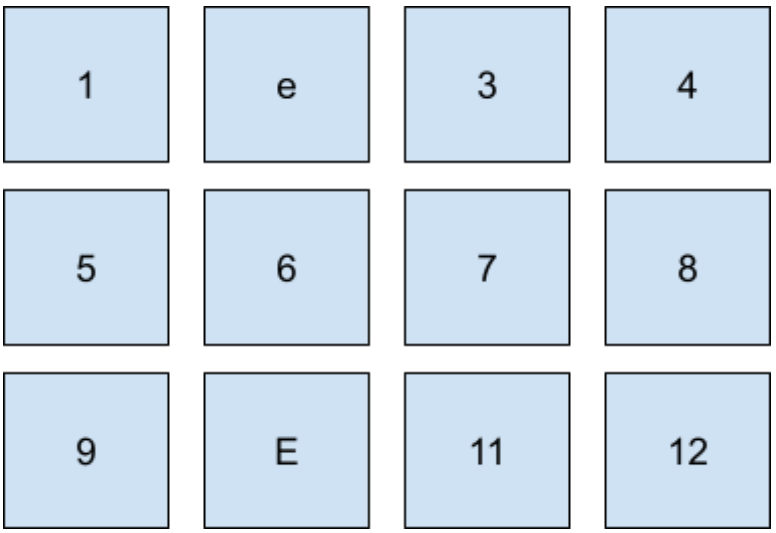


Seleccione un par: 1, 12

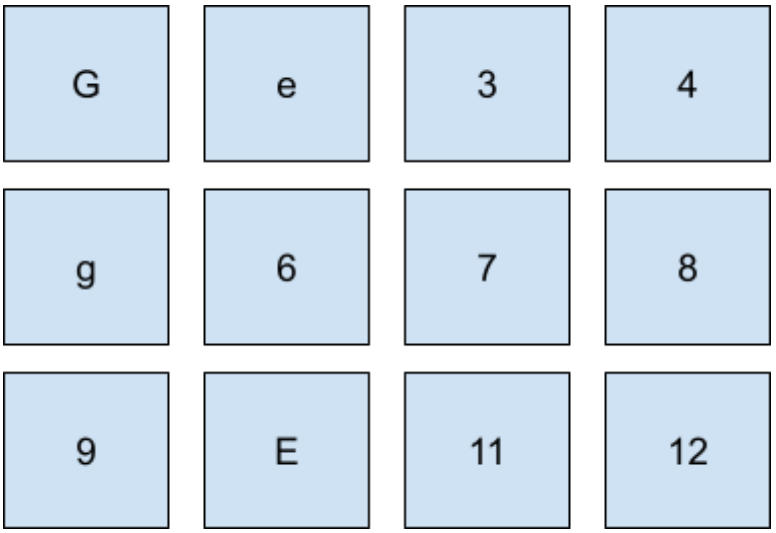


Sigue intentando.

Jugador : mdyagual
Pares vocales: 1
Pares consonantes: 0
Vidas: 1

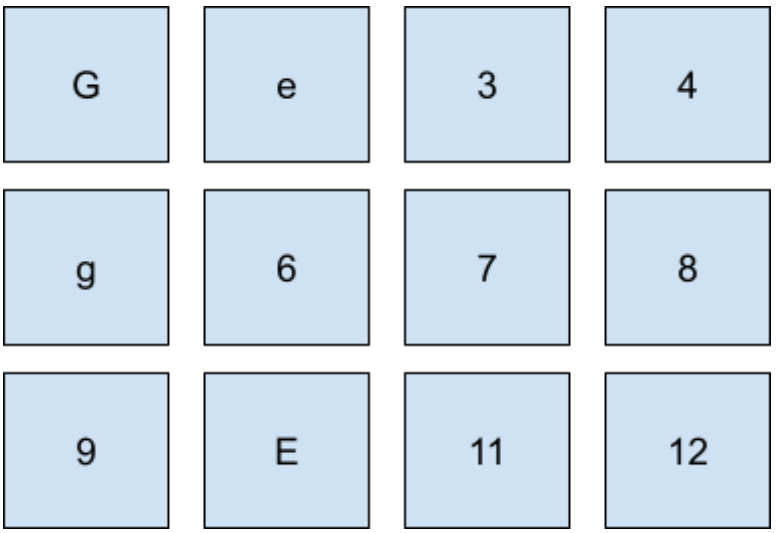


Seleccione un par: 5, 1

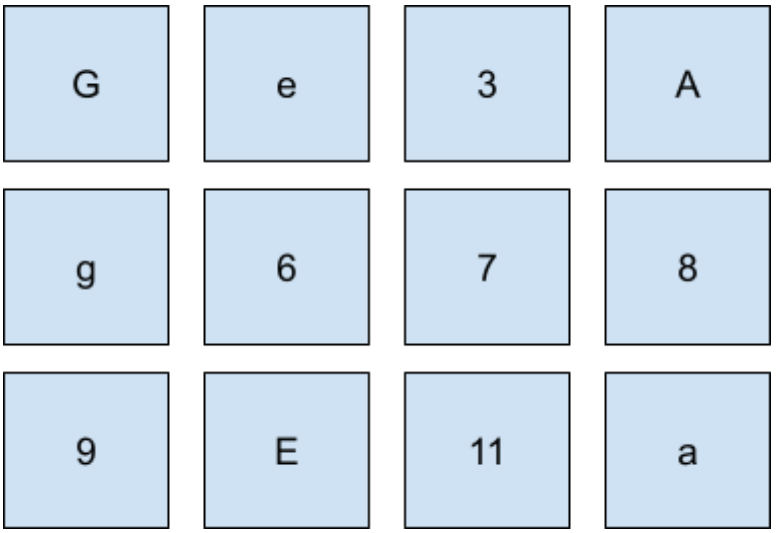


Has encontrado un par de consonantes.

Jugador : mdyagual
Pares vocales: 1
Pares consonantes: 1
Vidas: 1

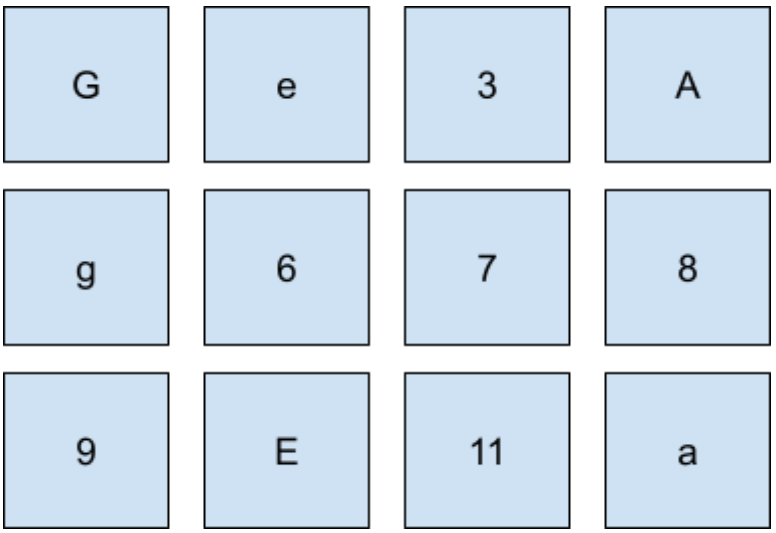


Seleccione un par: 12, 4

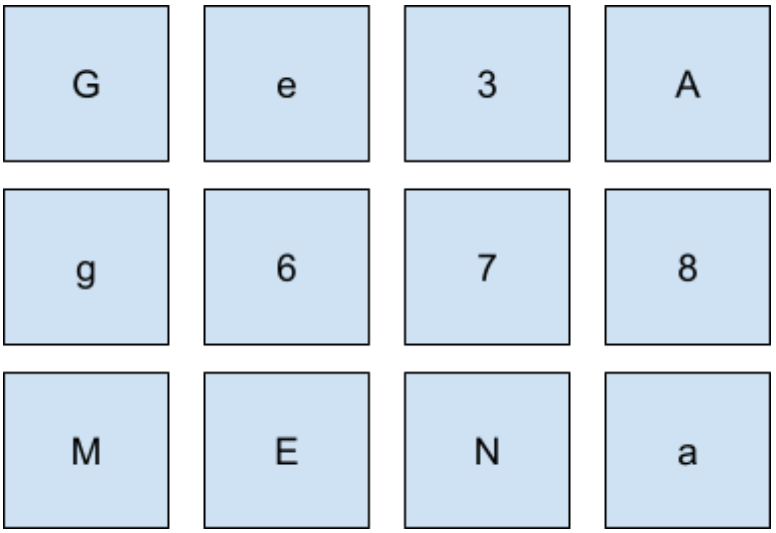


Has encontrado un par de vocales.

Jugador : mdyagual
Pares vocales: 2
Pares consonantes: 1
Vidas: 1



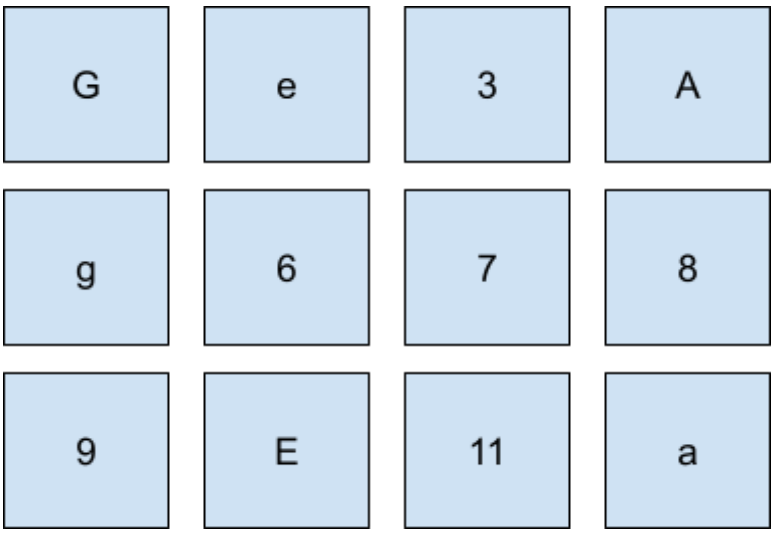
Seleccione un par: 9, 11



Sigue intentando.

Jugador : mdyagual
Pares vocales: 2
Pares consonantes: 1
Vidas: 0

Fin de la partida.



Preste atención a los cambios que sufre el tablero conforme avanzan las rondas, así mismo al puntaje y a las vidas.

Por hacer

memory_utils.ex

load_board(sel_letters)

Esta función tiene como objetivo ser el tablero resuelto, es decir contiene las coordenadas y pares correctos correspondientes a dichas coordenadas. sel_letters contiene los pares de letras correctos, solo falta generar las coordenadas correspondientes y asociarlas. HINTS:
https://hexdocs.pm/elixir/Enum.html#shuffle/1 ,
https://hexdocs.pm/elixir/Enum.html#chunk_every/2,
https://hexdocs.pm/elixir/Enum.html#map/2,
https://hexdocs.pm/elixir/Enum.html#zip/1
(2 puntos)

raw_positions(pos_pares, sel_letters)

Esta función tiene como objetivo lo mismo que load_board, sólo que de forma 'cruda'. Es decir cada letra asociada a una posición sin contar que si es vocal, consonante y tampoco si fue encontrada o no. Realizar el cambio 'raw' las posiciones y las letras a jugar correspondientes y asociarlas, generando un mapa. HINT:
https://hexdocs.pm/elixir/Enum.html#flat_map/2
(2 puntos)

memory_tasks.ex

selected_letters(map_alphabet)

Esta función llama a funciones de memory_utils.ex para armar las letras a jugar y devuelve una lista de tuplas que contienen las vocales y consonantes a considerar.

init_game()

Esta función es la encargada de dar inicio al juego, hace uso de la función anterior para poder luego llamar a load_board(sel_letters). Hay que solicitar un nickname al usuario e iniciar el juego considerando todo lo que necesita para comenzar. (1 punto)

game(board_on, solved_board, player, lifes, acc_v, acc_c)

Esta función es la encargada de iniciar y darle vida a la dinámica del juego, explicada anteriormente en el contexto llamando también a una función de memory_utils.ex. En ella se controla el conteo de vocales, consonantes, vidas, y finalización del juego. Adicionalmente hace el llamado a otra función para obtener la dinámica de revelado de cartas. Debe codificar lo necesario aplicando los principios de programación funcional. HINT: Case + Pattern matching, unless. (5 puntos)

reveal_cards(board_on, pair1, pair2)

Esta función es la encargada de revelar/volver a ocultar el par ingresado de acuerdo a las cordenadas.

Restricciones e indicaciones generales

1. Lo entregado está 100% operativo y lo que se solicita es para completar la funcionalidad del juego.
2. Puede crear funciones adicionales de ser necesario..
3. El uso de if está limitado a una única vez, si lo utiliza más veces será penalizado con reducción de puntos.
4. El uso de for está prohibido.
5. Considere los HINTS dados y no dude en consultar en la documentación.
6. Si edita lo entregado por el coach corre el riesgo de corromper la funcionalidad, a menos que sepa arreglarla o mejorarla.

Entregables

- Video de no más de 7 minutos que debe incluir (En este orden):
    - El juego funcionando 100%
    - Explicar en qué partes del código se encuentran los principios de programación funcional y por qué.
    - Explicar cómo completó la función game().
- Enlace a repositorio github con los dos archivos .ex.