Ejercicio #1

1. Datos de enemigos:

- Se define un arreglo llamado players que almacena objetos.
- Cada objeto representa a un enemigo e incluye propiedades:
    - name: Nombre del enemigo (ej: "enemyA").
    - distance: Distancia del enemigo (ej: 10).

2. Función findClosestEnemy:

- Toma un arreglo de enemigos (players) como parámetro.
- Inicializa una variable closestEnemy con el primer elemento de players.
- Recorre el arreglo desde la posición 1 (i = 1) hasta el final (i < players.length).
    - Para cada enemigo i (excepto el primero):
        - Si la distancia del enemigo actual (players[i].distance) es menor a la distancia del enemigo más cercano (closestEnemy.distance):
            - Actualiza closestEnemy con el enemigo actual.
- La función retorna el nombre del enemigo más cercano (closestEnemy.name).

3. Imprimiendo el resultado:

- Se imprime un mensaje que indica "El enemigo más cercano es:" seguido del nombre obtenido de la función findClosestEnemy.

Pseudocodigo:
Entradas: Arreglo de enemigos (`players`)

Salida: Nombre del enemigo más cercano

Inicializar `closestEnemy` con el primer elemento de `players`

Para cada enemigo desde el segundo (`i = 1`) hasta el final del arreglo:
  Si la distancia del enemigo actual es menor a la distancia de `closestEnemy`
    Actualizar `closestEnemy` con el enemigo actual

Retornar el nombre de `closestEnemy`

Imprimir mensaje con el nombre del enemigo más cercano obtenido de la función

Ejercicio #2

1. Datos de enemigos:

- Se define un arreglo llamado enemies que almacena objetos.
- Cada objeto representa a un enemigo e incluye propiedades:
    - name: Nombre del enemigo (ej: "enemyX").
    - distance: Distancia del enemigo (ej: 15).

2. Función findClosestEnemy:

- Toma un arreglo de enemigos (enemies) como parámetro.
- Inicializa una variable closestEnemy con el primer elemento de enemies.
- Recorre el arreglo desde la posición 1 (i = 1) hasta el final (i < enemies.length).
    - Para cada enemigo i (excepto el primero):
        - Si la distancia del enemigo actual (enemies[i].distance) es menor a la distancia del enemigo más cercano (closestEnemy.distance):
            - Actualiza closestEnemy con el enemigo actual.
        - Error: Actualmente, el código también actualiza closestEnemy si la distancia es igual (===) sin ninguna diferenciación. Esto podría seleccionar un enemigo aleatorio con la misma distancia en lugar del primero encontrado.

3. Imprimiendo el resultado:

- La función retorna el nombre del enemigo más cercano (closestEnemy.name).
- Fuera de la función:
    - Se llama a la función findClosestEnemy para obtener el nombre del enemigo más cercano y se almacena en closestEnemyName.
    - Se imprime un mensaje que indica "El enemigo que dispara es:" seguido del nombre obtenido.

Pseudocodigo:
Entradas: Arreglo de enemigos (`enemies`)

Salida: Nombre del enemigo más cercano (con error en selección de distancia igual)

Inicializar `closestEnemy` con el primer elemento de `players`

Para cada enemigo desde el segundo (`i = 1`) hasta el final del arreglo:

Si la distancia del enemigo actual es menor a la distancia de `closestEnemy`
Actualizar `closestEnemy` con el enemigo actual
Si la distancia del enemigo actual es igual a la distancia de `closestEnemy` (**Error, no prioriza el primero con distancia igual**)
Actualizar `closestEnemy` con el enemigo actual (selecciona aleatoriamente entre enemigos con la misma distancia)

Retornar el nombre de `closestEnemy`

Obtener nombre del enemigo más cercano llamando a `findClosestEnemy` y almacenar en `closestEnemyName`

Imprimir mensaje con el nombre del enemigo obtenido

Ejercicio #3

1. Almacenamiento de enemigos:

- Se declara una variable enemies como un arreglo vacío.
- Este arreglo se utilizará para almacenar información sobre los enemigos detectados.

2. Actualizando información de enemigos:

- La función updateEnemy toma dos parámetros:
  - enemyName: Nombre del enemigo (ej: "enemigoX").
  - newDistance: Nueva distancia del enemigo.
- Busca la posición (enemyIndex) del enemigo en el arreglo enemies usando la función findIndex y el nombre del enemigo.
  - Si el enemigo se encuentra (enemyIndex !== -1):
    - Actualiza la propiedad distance del enemigo en la posición encontrada con la nueva distancia.
  - Si el enemigo no se encuentra (enemyIndex === -1):
    - Crea un nuevo objeto con las propiedades name (nombre del enemigo) y distance (nueva distancia) y lo agrega al final del arreglo enemies.

3. Encontrar al enemigo más cercano:

- La función findClosestEnemy no recibe parámetros.
- Inicializa una variable closestEnemy con el primer elemento del arreglo enemies (considerando que al menos hay un enemigo).
- Recorre el arreglo enemies desde la posición 1 (i = 1) hasta el final (i < enemies.length).
    - Para cada enemigo i (excepto el primero):
        - Si la distancia del enemigo actual (enemies[i].distance) es menor a la distancia del enemigo más cercano (closestEnemy.distance):
            - Actualiza closestEnemy con el enemigo actual (nuevo enemigo más cercano).
- La función retorna el nombre del enemigo más cercano (closestEnemy.name).

4. Bucle continuo de actualización y búsqueda:

- Un bucle while(true) se ejecuta infinitamente.
    - Solicita información del enemigo 1: nombre y distancia usando prompt.
    - Convierte la distancia del enemigo 1 a un número usando parseFloat.
    - Repite el mismo proceso para solicitar información del enemigo 2.
    - Llama a la función updateEnemy dos veces, actualizando la información del enemigo 1 y 2 en el arreglo enemies.
    - Llama a la función findClosestEnemy para obtener el nombre del enemigo más cercano.
    - Imprime un mensaje con el nombre del enemigo más cercano obtenido.

Pseudocodigo:

Inicializar un arreglo vacío enemies para almacenar enemigos.
Función updateEnemyEntradas: enemyName (nombre del enemigo), newDistance (nueva distancia) Buscar la posición (enemyIndex) del enemigo en enemies por su nombre.Si el enemigo se encuentra (enemyIndex !== -1):Actualizar la propiedad distance del enemigo en enemies[enemyIndex] con newDistance.Si el enemigo no se encuentra (enemyIndex === -1):Crear un nuevo objeto con name y distance y agregarlo al final de enemies.

Función findClosestEnemyInicializar closestEnemy con el primer elemento de enemies.Para cada enemigo desde el segundo (i = 1) hasta el final del arreglo:Si la distancia del enemigo actual es menor a la distancia de closestEnemy:Actualizar closestEnemy con el enemigo actual.Retornar el nombre de closestEnemy.

Bucle infinito:Mientras sea verdadero (while(true)): - Solicitar y almacenar nombre y distancia del enemigo 1. - Convertir la distancia del enemigo 1 a un número. - Repetir el proceso para el enemigo 2. - Actualizar información de enemigos 1 y 2 llamando a updateEnemy. - Encontrar el enemigo más cercano llamando a findClosestEnemy y almacenar su nombre en closestEnemyName. - Imprimir mensaje con el nombre del enemigo más cercano (closestEnemyName).

Ejercicio #4

1. Almacenamiento de enemigos:

- Se declara una variable enemies como un arreglo vacío.
- Este arreglo se utilizará para almacenar información sobre los enemigos detectados, incluyendo su nombre, distancia y prioridad.

2. Actualizando información de enemigos:

- La función updateEnemy toma tres parámetros:
    - enemyName: Nombre del enemigo (ej: "enemigoX").
    - newDistance: Nueva distancia del enemigo.
    - newPriority: Nueva prioridad del enemigo (mayor número, mayor prioridad).
- Busca la posición (enemyIndex) del enemigo en el arreglo enemies usando la función findIndex y el nombre del enemigo.
    - Si el enemigo se encuentra (enemyIndex !== -1):
        - Actualiza la propiedad distance y priority del enemigo en la posición encontrada con los nuevos valores.
    - Si el enemigo no se encuentra (enemyIndex === -1):
        - Crea un nuevo objeto con las propiedades name (nombre del enemigo), distance (nueva distancia) y priority (nueva prioridad) y lo agrega al final del arreglo enemies.

3. Encontrar al enemigo con mayor prioridad:

- La función findHighestPriorityEnemy no recibe parámetros.
- Inicializa una variable highestPriorityEnemy con el valor null (indica que aún no se ha encontrado un enemigo con prioridad).
- Recorre el arreglo enemies usando un bucle for...of.

    - Para cada enemigo:
        - Si highestPriorityEnemy es null o la prioridad del enemigo actual (enemy.priority) es mayor a la del enemigo con mayor prioridad actual:
            - Actualiza highestPriorityEnemy con el enemigo actual (nuevo enemigo con mayor prioridad).
        - Si las prioridades son iguales (enemy.priority === highestPriorityEnemy.priority):
            - Si la distancia del enemigo actual (enemy.distance) es menor a la del enemigo con mayor prioridad actual:
                - Actualiza highestPriorityEnemy con el enemigo actual (prioridad igual, pero más cercano).
- La función retorna el nombre del enemigo con mayor prioridad (highestPriorityEnemy.name) si se encontró alguno, o null si no hay enemigos.

4. Bucle continuo de actualización y búsqueda:

- Un bucle while(true) se ejecuta infinitamente.
    - Solicita información del enemigo 1: nombre, distancia y prioridad usando prompt.
    - Convierte la distancia y prioridad del enemigo 1 a números usando parseFloat y parseInt respectivamente.
    - Repite el mismo proceso para solicitar información del enemigo 2.
    - Llama a la función updateEnemy dos veces, actualizando la información del enemigo 1 y 2 en el arreglo enemies.
    - Llama a la función findHighestPriorityEnemy para obtener el nombre del enemigo con mayor prioridad.
    - Imprime un mensaje con el nombre del enemigo con mayor prioridad obtenido.

Pseudocodigo:

Inicializar un arreglo vacío enemies para almacenar enemigos.
Función updateEnemyEntradas: enemyName (nombre del enemigo), newDistance (nueva distancia), newPriority (nueva prioridad) Buscar la posición (enemyIndex) del enemigo en enemies por su nombre.Si el enemigo se encuentra (enemyIndex !== -1):Actualizar las propiedades distance y priority del enemigo en enemies[enemyIndex] con newDistance y newPriority respectivamente.Si el enemigo no se encuentra (enemyIndex === -1):Crear un nuevo objeto con name, distance y priority y agregarlo al final de enemies.

Función findHighestPriorityEnemyInicializar highestPriorityEnemy con null.Para cada enemigo en enemies:Si highestPriorityEnemy es null o la prioridad del enemigo actual es mayor a la de highestPriorityEnemy:Actualizar highestPriorityEnemy con el enemigo actual.Si las prioridades son iguales:Si la distancia del enemigo actual es menor a la de highestPriorityEnemy:Actualizar highestPriorityEnemy con el enemigo actual.Retornar el nombre de highestPriorityEnemy (o null si no hay enemigos).

Ejercicio #4

1. Almacenamiento de enemigos:

- Se declara una variable enemies como un arreglo vacío.
- Este arreglo se utilizará para almacenar información sobre los enemigos detectados, incluyendo su nombre, distancia, prioridad y velocidad.

2. Actualizando información de enemigos:

- La función updateEnemy toma cuatro parámetros:
    - enemyName: Nombre del enemigo (ej: "enemigoX").
    - newDistance: Nueva distancia del enemigo.
    - newPriority: Nueva prioridad del enemigo (mayor número, mayor prioridad).
    - newSpeed: Nueva velocidad del enemigo.
- Busca la posición (enemyIndex) del enemigo en el arreglo enemies usando la función findIndex y el nombre del enemigo.
    - Si el enemigo se encuentra (enemyIndex !== -1):
        - Actualiza las propiedades distance, priority y speed del enemigo en la posición encontrada con los nuevos valores.
    - Si el enemigo no se encuentra (enemyIndex === -1):
        - Crea un nuevo objeto con las propiedades name (nombre del enemigo), distance (nueva distancia), priority (nueva prioridad) y speed (nueva velocidad) y lo agrega al final del arreglo enemies.

3. Encontrar al enemigo con mayor prioridad, más cercano y más rápido:

- La función findHighestPriorityClosestFastestEnemy no recibe parámetros.

- Inicializa una variable highestPriorityEnemy con el valor null (indica que aún no se ha encontrado un enemigo que cumpla las condiciones).
- Recorre el arreglo enemies usando un bucle for...of.
    - Para cada enemigo:
        - Si highestPriorityEnemy es null o la prioridad del enemigo actual (enemy.priority) es mayor a la del enemigo con mayor prioridad actual:
            - Actualiza highestPriorityEnemy con el enemigo actual (nuevo enemigo con mayor prioridad).
        - Si las prioridades son iguales (enemy.priority === highestPriorityEnemy.priority):
            - Si la distancia del enemigo actual (enemy.distance) es menor a la del enemigo con mayor prioridad actual:
                - Actualiza highestPriorityEnemy con el enemigo actual (prioridad igual, pero más cercano).
            - Si las distancias son iguales (enemy.distance === highestPriorityEnemy.distance):
                - Si la velocidad del enemigo actual (enemy.speed) es mayor a la del enemigo con mayor prioridad actual:
                    - Actualiza highestPriorityEnemy con el enemigo actual (prioridad e igual distancia, pero más rápido).
- La función retorna el nombre del enemigo que cumple las condiciones (highestPriorityEnemy.name) si se encontró alguno, o null si no hay enemigos.

4. Bucle continuo de actualización y búsqueda:

- Un bucle while(true) se ejecuta infinitamente.
    - Solicita información del enemigo 1: nombre, distancia, prioridad y velocidad usando prompt.
    - Convierte la distancia, prioridad y velocidad del enemigo 1 a números usando parseFloat y parseInt respectivamente.
    - Repite el mismo proceso para solicitar información del enemigo 2.
    - Llama a la función updateEnemy dos veces, actualizando la información del enemigo 1 y 2 en el arreglo enemies.
    - Llama a la función findHighestPriorityClosestFastestEnemy para obtener el nombre del enemigo con mayor prioridad, más cercano y más rápido.
    - Imprime un mensaje con el nombre del enemigo obtenido.

Pseudocodigo:

Inicializar un arreglo vacío enemies para almacenar enemigos.
Función updateEnemyEntradas: enemyName (nombre del enemigo), newDistance (nueva distancia), newPriority (nueva prioridad), newSpeed (nueva velocidad) Buscar la posición (enemyIndex) del enemigo en enemies por su nombre.Si el enemigo se encuentra (enemyIndex !== -1):Actualizar las propiedades distance, priority y speed del enemigo en enemies[enemyIndex] con newDistance, newPriority y newSpeed respectivamente.Si el enemigo no se encuentra (enemyIndex === -1):Crear un nuevo objeto con name, distance, priority y speed y agregarlo al final de enemies.
Función findHighestPriorityClosestFastestEnemyInicializar highestPriorityEnemy con null.

Función findHighestPriorityClosestFastestEnemy (continuación)Para cada enemigo en enemies: - Si highestPriorityEnemy es null o la prioridad del enemigo actual (enemy.priority) es mayor a la del enemigo con mayor prioridad actual: - Actualizar highestPriorityEnemy con el enemigo actual (nuevo enemigo con mayor prioridad). - Si las prioridades son iguales (enemy.priority === highestPriorityEnemy.priority): - Si la distancia del enemigo actual (enemy.distance) es menor a la del enemigo con mayor prioridad actual: - Actualizar highestPriorityEnemy con el enemigo actual (prioridad igual, pero más cercano). - Si las distancias son iguales (enemy.distance === highestPriorityEnemy.distance): - Si la velocidad del enemigo actual (enemy.speed) es mayor a la del enemigo con mayor prioridad actual: - Actualizar highestPriorityEnemy con el enemigo actual (prioridad e igual distancia, pero más rápido).Retornar el nombre de highestPriorityEnemy (o null si no hay enemigos).

Bucle infinito: Mientras sea verdadero (while(true)): - Solicitar información del enemigo 1: nombre, distancia, prioridad y velocidad usando prompt. - Convertir la distancia, prioridad y velocidad del enemigo 1 a números usando parseFloat y parseInt respectivamente. - Repetir el mismo proceso para solicitar información del enemigo 2. - Llamar a la función updateEnemy dos veces, actualizando la información del enemigo 1 y 2 en el arreglo enemies. - Llamar a la función findHighestPriorityClosestFastestEnemy para obtener el nombre del enemigo con mayor prioridad, más cercano y más rápido. - Imprimir un mensaje con el nombre del enemigo obtenido.