Problema I (15 puntos) Escribe un programa que lea continuamente el nombre y la distancia de dos enemigos y determine cuál de ellos está más cerca. Imprime el nombre del enemigo más cercano.

```
const enemigos = [
    { name: 'EnemyA', distance: 10 },
    { name: 'EnemyB', distance: 20 },
];

function encontrarCercano(enemigos) {
    let enemigoCercano = enemigos[0];
    for (let i = 1; i < enemigos.length; i++) {
        if (enemigos[i].distance < enemigoCercano.distance) {
            enemigoCercano = enemigos[i];
        }
    }
    return enemigoCercano.name;
}

console.log("El enemigo más cercano es: " + encontrarCercano(enemigos));
```

// Definimos un array de objetos llamado 'enemigos' que contiene dos enemigos con sus respectivas distancias

// Definimos una función llamada 'encontrarCercano' que toma como argumento un array de enemigos

// Inicializamos la variable 'enemigoCercano' con el primer enemigo del array

// Recorremos el array de enemigos comenzando desde el segundo elemento (índice 1)

// Comparamos la distancia del enemigo actual con la distancia del 'enemigoCercano'

// Si la distancia del enemigo actual es menor, actualizamos 'enemigoCercano' con el enemigo actual

// Retornamos el nombre del enemigo más cercano

// Llamamos a la función 'encontrarCercano' con el array 'enemigos' y mostramos el resultado en la consola

// Resultado en consola: "El enemigo más cercano es: EnemyA"

Problema II (15 puntos) En un juego de disparos, tu objetivo es atacar siempre al enemigo más cercano. Escribe un programa que lea continuamente el nombre y la distancia de dos enemigos y seleccione el objetivo más cercano para atacar. Si ambos enemigos están a la misma distancia, selecciona el que aparece primero.

```
1   const enemigos = [
2       { name: 'EnemyX', distance: 10 },
3       { name: 'EnemyY', distance: 10 },
4   ];
5
6   function encontrarCercano(enemigos) {
7       let enemigoCercano = enemigos[0];
8       for (let i = 1; i < enemigos.length; i++) {
9           // Solo actualizamos si encontramos una distancia menor,
10          // lo que significa que si las distancias son iguales,
11          // enemigoCercano no cambiará y se mantendrá el primero.
12          if (enemigos[i].distance < enemigoCercano.distance) {
13              enemigoCercano = enemigos[i];
14          }
15      }
16      return enemigoCercano.name;
17  }
18
19  console.log("El enemigo más cercano es: " + encontrarCercano(enemigos));
20
```

// Definimos un array de objetos llamado 'enemigos' que contiene dos enemigos con sus respectivas distancias

// Definimos una función llamada 'encontrarCercano' que toma como argumento un array de enemigos

// Inicializamos la variable 'enemigoCercano' con el primer enemigo del array

// Recorremos el array de enemigos comenzando desde el segundo elemento (índice 1)

// Solo actualizamos si encontramos una distancia menor,
// lo que significa que si las distancias son iguales,
// 'enemigoCercano' no cambiará y se mantendrá el primero.

// Retornamos el nombre del enemigo más cercano

// Llamamos a la función 'encontrarCercano' con el array 'enemigos' y mostramos el resultado en la consola

// Resultado en consola: "El enemigo más cercano es: EnemyX"

Problema III (20 Puntos) En una simulación de radar, los datos de enemigos y sus distancias cambian dinámicamente. Escribe un programa que lea continuamente el nombre y la distancia de dos enemigos y determine cuál de ellos está más cerca. Asegúrate de que el programa maneje correctamente entradas donde las distancias pueden cambiar rápidamente. Utiliza prompt para ingresar los datos.

```
while (true) {
    let name1 = prompt("Ingrese el nombre del primer enemigo:");
    if (name1 === null) break;
    let distance1 = parseFloat(prompt("Ingrese la distancia del primer enemigo:"));
    if (distance1 === null || distance1 < 0) break;

    let name2 = prompt("Ingrese el nombre del segundo enemigo:");
    if (name2 === null) break;
    let distance2 = parseFloat(prompt("Ingrese la distancia del segundo enemigo:"));
    if (distance2 === null || distance2 < 0) break;

    let enemigoCercano;
    if (distance1 < distance2) {
        enemigoCercano = name1;
    } else if (distance2 < distance1) {
        enemigoCercano = name2;
    } else {
        enemigoCercano = name1;
    }

    console.log("El enemigo más cercano es: " + enemigoCercano  );
}

```

// Iniciamos un bucle infinito que permitirá leer datos de entrada continuamente

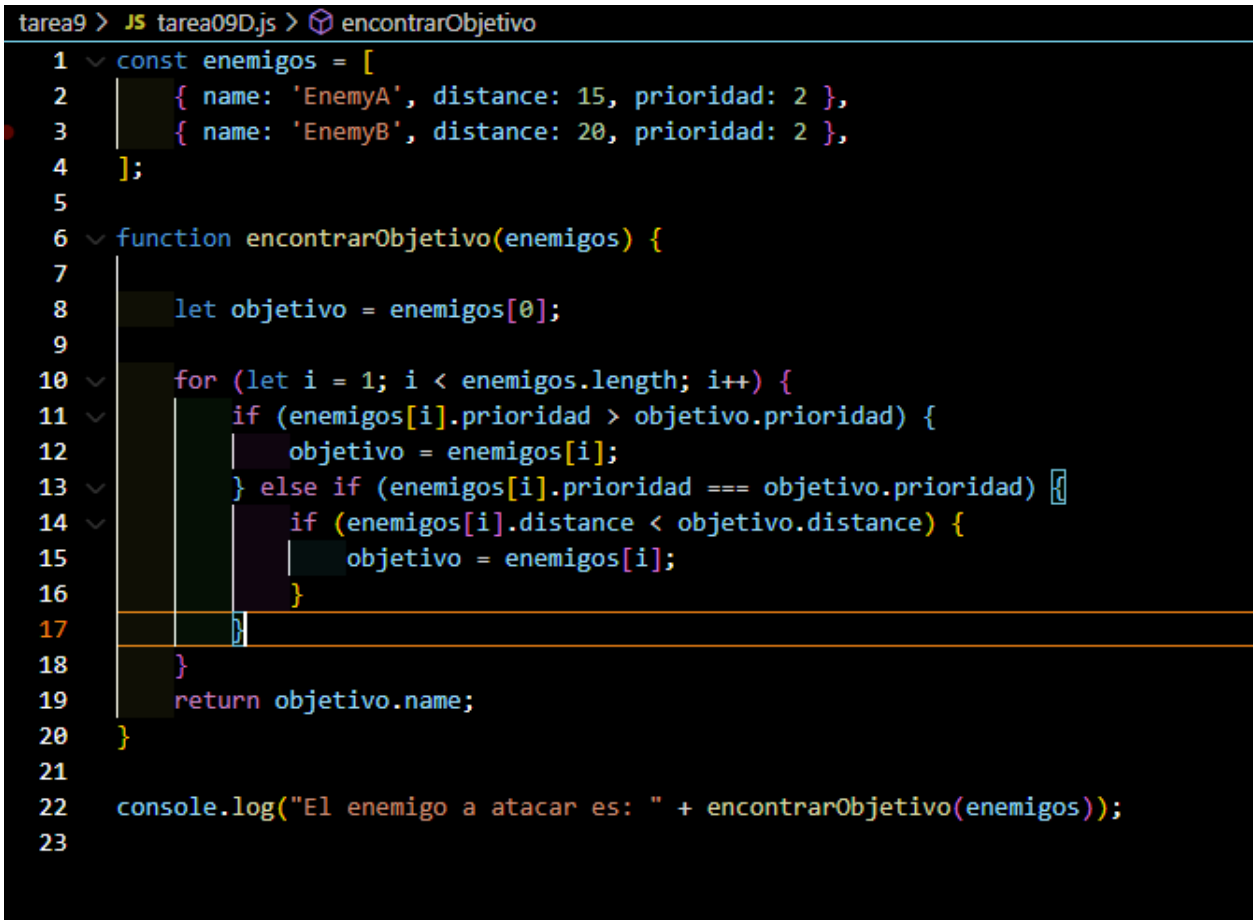
// Solicitamos al usuario que ingrese el nombre del primer enemigo
// Si el usuario cancela el prompt, salimos del bucle
// Solicitamos al usuario que ingrese la distancia del primer enemigo
// Si el usuario cancela el prompt o ingresa una distancia negativa, salimos del bucle

// Solicitamos al usuario que ingrese el nombre del segundo enemigo
// Si el usuario cancela el prompt, salimos del bucle
// Solicitamos al usuario que ingrese la distancia del segundo enemigo
// Si el usuario cancela el prompt o ingresa una distancia negativa, salimos del bucle

// Inicializamos la variable 'enemigoCercano' para determinar cuál enemigo está más cerca
// Comparamos las distancias de los dos enemigos
// Si la distancia del primer enemigo es menor, actualizamos 'enemigoCercano' con el nombre del primer enemigo
// Si la distancia del segundo enemigo es menor, actualizamos 'enemigoCercano' con el nombre del segundo enemigo
// Si las distancias son iguales, seleccionamos el primer enemigo ingresado

// Mostramos en la consola el nombre del enemigo más cercano

Problema IV (20 Puntos) En una situación de combate, además de la distancia, se te proporciona la prioridad de ataque de cada enemigo. Escribe un programa que lea continuamente el nombre, la distancia y la prioridad de dos enemigos y seleccione el enemigo con mayor prioridad para atacar. Si ambos enemigos tienen la misma prioridad, selecciona el más cercano.



// Definimos un array de objetos llamado 'enemigos' que contiene dos enemigos con sus respectivas distancias y prioridades

// Definimos una función llamada 'encontrarObjetivo' que toma como argumento un array de enemigos

// Inicializamos la variable 'objetivo' con el primer enemigo del array

// Recorremos el array de enemigos comenzando desde el segundo elemento (índice 1)

// Comparamos la prioridad del enemigo actual con la prioridad del 'objetivo'
// Si la prioridad del enemigo actual es mayor, actualizamos 'objetivo' con el enemigo actual

// Si las prioridades son iguales, comparamos la distancia del enemigo actual con la distancia del 'objetivo'
// Si la distancia del enemigo actual es menor, actualizamos 'objetivo' con el enemigo actual

// Retornamos el nombre del enemigo objetivo

// Llamamos a la función 'encontrarObjetivo' con el array 'enemigos' y mostramos el resultado en la consola

// Resultado en consola: "El enemigo a atacar es: EnemyA"

Problema V (30 puntos) Desarrolla un programa de simulación de radar que lea continuamente el nombre, la distancia, la velocidad y la prioridad de ataque de dos enemigos. El programa debe seleccionar el enemigo con mayor prioridad para atacar. Si ambos tienen la misma prioridad, selecciona el más cercano. Si las distancias son iguales, selecciona el enemigo con mayor velocidad.

// Definimos un array de objetos llamado 'enemigos' que contiene dos enemigos con sus respectivas distancias, prioridades y velocidades

// Definimos una función llamada 'encontrarObjetivo' que toma como argumento un array de enemigos

// Inicializamos la variable 'objetivo' con el primer enemigo del array

// Recorremos el array de enemigos comenzando desde el segundo elemento (índice 1)

// Comparamos la prioridad del enemigo actual con la prioridad del 'objetivo'
// Si la prioridad del enemigo actual es mayor, actualizamos 'objetivo' con el enemigo actual

// Si las prioridades son iguales, comparamos la distancia del enemigo actual con la distancia del 'objetivo'
// Si la distancia del enemigo actual es menor, actualizamos 'objetivo' con el enemigo actual

// Si las distancias son iguales, comparamos la velocidad del enemigo actual con la velocidad del 'objetivo'
// Si la velocidad del enemigo actual es mayor, actualizamos 'objetivo' con el enemigo actual

// Retornamos el nombre del enemigo objetivo

// Llamamos a la función 'encontrarObjetivo' con el array 'enemigos' y mostramos el resultado en la consola

// Resultado en consola: "El enemigo a atacar es: EnemyB"