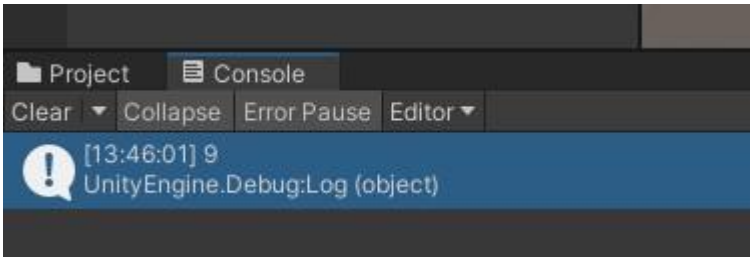
Tarea 1

Ejercicio 1:
Implementar una función que recibe una lista de enteros L y un número entero n de forma que modifique la lista mediante el borrado de todos los elementos de la lista que tengan este valor.

En la primera parte hice una lista con 10 elementos, coloque un if donde si la variable int N era igual a 5 entonces L eliminaría de su lista a su elemento 5 y en la consola me imprime que la lista tiene 9 elementos.

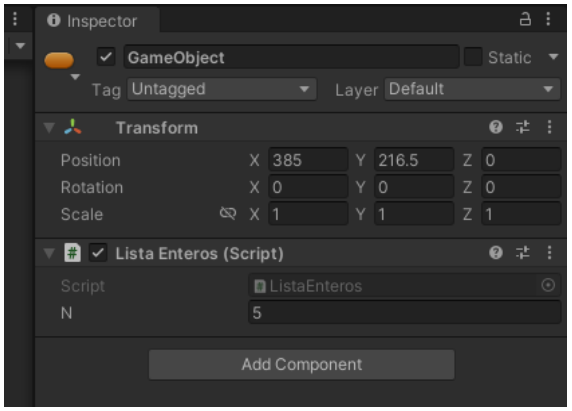


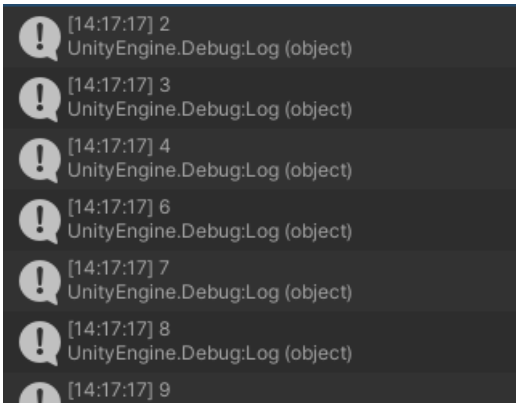

```
List<int> L = new List<int>();
    public int N;
    // Start is called before the first frame update
    void Start()
    {
        L.Add(1);
        L.Add(2);
        L.Add(3);
        L.Add(4);
        L.Add(5);
        L.Add(6);
        L.Add(7);
        L.Add(8);
        L.Add(9);
        L.Add(10);

        if(N == 5)
        {
            L.Remove(5);
            Debug.Log(L.Count);
        }
    }
```

Si el if lo meto en un método también me imprime solo 9 elementos.

```
List<int> L = new List<int>();
   public int N;

   void Start()
   {
      L.Add(1);
      L.Add(2);
      L.Add(3);
      L.Add(4);
      L.Add(5);
      L.Add(6);
      L.Add(7);
      L.Add(8);
      L.Add(9);
      L.Add(10);

      Borrado();
   }


   void Borrado()
   {
      if(N == 5)
      {
         L.Remove(5);
         Debug.Log(L.Count);
      }
   }
```

Y si quiero ver los valores de la lista en consola, utilice un foreach en el método y sale los elementos menos el que es parecido a N.



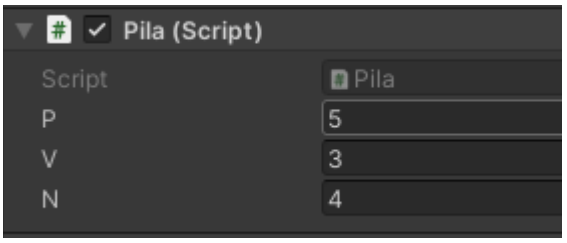


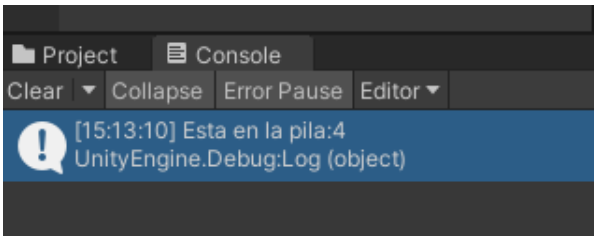
```
List<int> L = new List<int>();
    public int N;

    void Start()
    {
        L.Add(1);
        L.Add(2);
        L.Add(3);
        L.Add(4);
        L.Add(5);
        L.Add(6);
        L.Add(7);
        L.Add(8);
        L.Add(9);
        L.Add(10);

        Borrado();
    }


    void Borrado()
    {
        if(N == 5)
        {
            L.Remove(5);

            foreach (int numeros in L)
            {
                Debug.Log(numeros);
            }
        }

    }
```

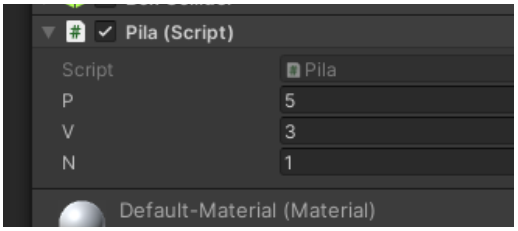
Ejercicio 2:
Escribir una función Reemplazar que tenga como argumentos una pila con tipo de elemento int y dos valores int: nuevo y viejo, de forma que si el segundo valor aparece en algún lugar de la pila, sea reemplazado por el primero.

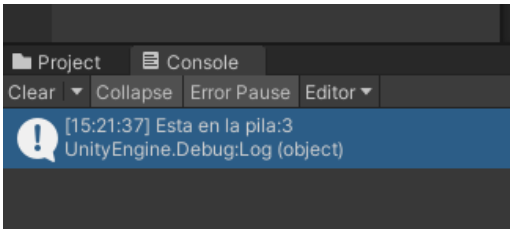
Hice 3 variables int (P,N, V) y también métodos diferentes para cada if (si el nuevo número es mayor al viejo, si es menor que el viejo, si es igual a la pila o incluso si es mayor que la pila y no está).

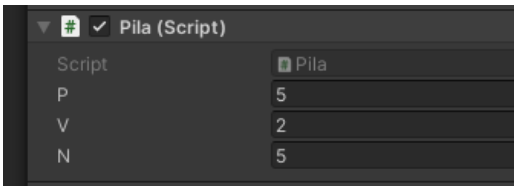
Pregunta: ¿Cómo puedo cancelar un método para que no se repita como en el caso 3 y 4?

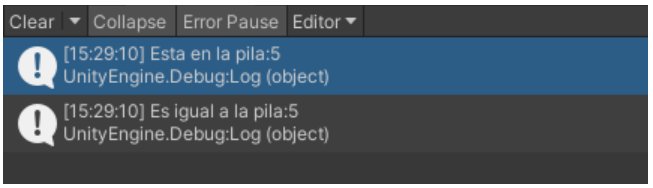


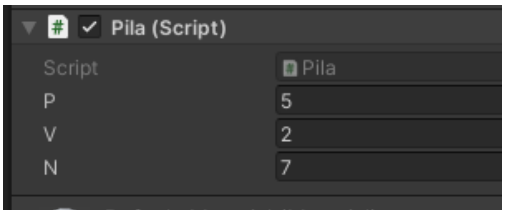


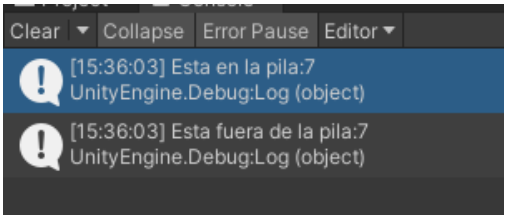










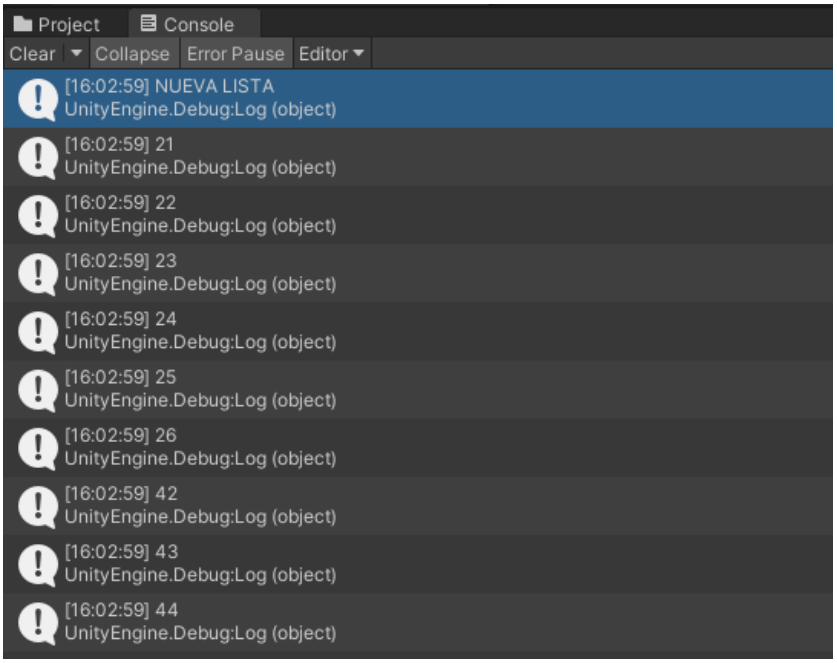



```
public int P;
    public int V;
    public int N;

    void Start()
    {
        if(N > V)
        {
```

```
        Reemplazar();
    }

    if(N < V)
    {
        Inverso();
    }

    if(N == P)
    {
        Igual();
    }

    if(N > P)
    {
        Fuera();
    }
}


void Reemplazar()
{
    Debug.Log("Esta en la pila:" + N);
}

void Inverso()
{
    Debug.Log("Esta en la pila:" + V);
}

void Igual()
{
    Debug.Log("Es igual a la pila:" + P);
}

void Fuera()
{
    Debug.Log("Esta fuera de la pila:" + N);
}
```

Ejercicio 3:
Implementar una función Mezcla2 que tenga como parámetros dos listas de enteros ordenados de menor a mayor y que devuelva una nueva lista como unión de ambas con sus elementos ordenados de la misma forma.

Utilice la misma fórmula del ejercicio 1 solo que cambie el nombre y los elementos de cada lista y cree un método que me imprime los datos de ambas listas como una suma.

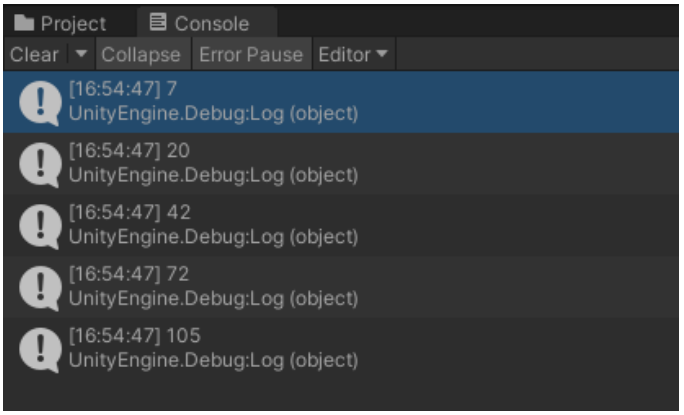



```
List<int> Lista1 = new List<int>();
   List<int> Lista2 = new List<int>();

   void Start()
   {
      Lista1.Add(21);
      Lista1.Add(22);
      Lista1.Add(23);
      Lista1.Add(24);
      Lista1.Add(25);
      Lista1.Add(26);

      Lista2.Add(42);
      Lista2.Add(43);
```

```
    Lista2.Add(44);
    Lista2.Add(45);
    Lista2.Add(46);
    Lista2.Add(47);

    Debug.Log("NUEVA LISTA");
    Mezcla2();
}


void Mezcla2()
{
    foreach (int numeros1 in Lista1)
    {
        Debug.Log(numeros1);
    }

    foreach (int numeros2 in Lista2)
    {
        Debug.Log(numeros2);
    }

}
```

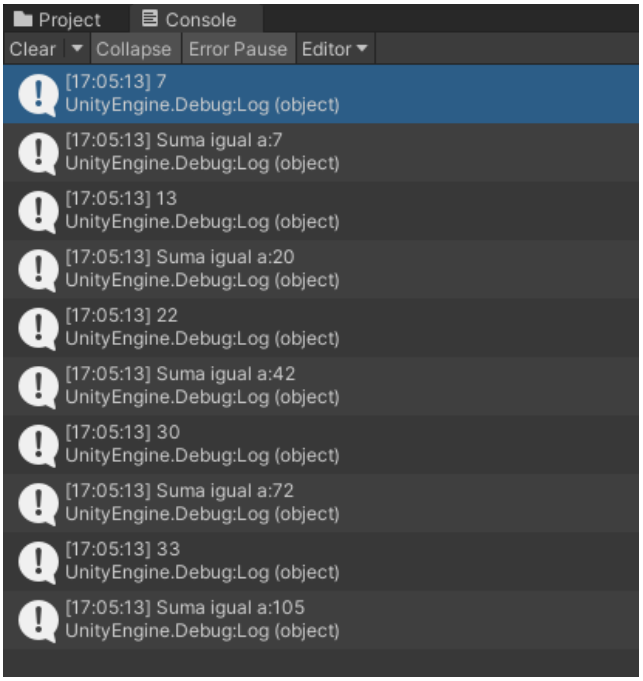
Ejercicio 4:
Construir una función que sume los elementos de una lista de enteros recursivamente.

Con la primera opción me salieron los datos y la suma total al final.




```
List<int> numeros = new List<int>();
    public int mas;

    void Start()
    {
        numeros.Add(7);
        numeros.Add(13);
        numeros.Add(22);
        numeros.Add(30);
        numeros.Add(33);

        Sumarnumeros();
    }


    void Sumarnumeros()
    {
        foreach (int n in numeros)
        {
            mas = mas + n;
            Debug.Log(mas);
        }

    }
```

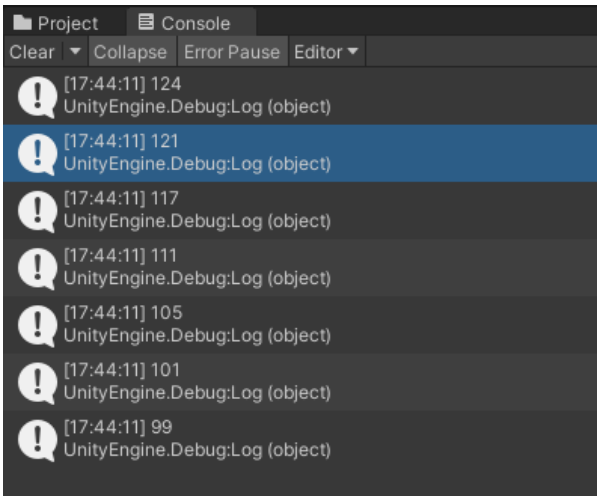
Y con la segunda sale la suma uno por uno y la final




```
List<int> numeros = new List<int>();
    public int mas;

    void Start()
    {
        numeros.Add(7);
        numeros.Add(13);
        numeros.Add(22);
        numeros.Add(30);
        numeros.Add(33);

        Sumarnumeros();
    }


    void Sumarnumeros()
    {
        foreach (int n in numeros)
        {
            Debug.Log(n);
            mas = mas + n;
            Debug.Log("Suma igual a:" + mas);
        }

    }
```

Ejercicio 5:
Construir una función que imprima los elementos de una lista enlazada de enteros en orden inverso a partir de una posición p.

Cree la lista y el método me dice que contará la lista al revés y me imprimira los elementos




```
List<int> lista = new List<int>();

    void Start()
    {
        lista.Add(99);
        lista.Add(101);
        lista.Add(105);
        lista.Add(111);
        lista.Add(117);
        lista.Add(121);
        lista.Add(124);

        Inverso();
    }


    void Inverso()
    {
        lista.Reverse();
        foreach (int l in lista)
        {
            Debug.Log(l);
        }
    }
```

Tablas HASH

Las tablas hash son un tipo de estructura de datos que sirve para clasificar los datos y poder acceder a ellos, buscar, ordenar o modificar de una forma más óptima.

Por ejemplo: utilizaremos “HashMap”, queremos guardar las partes de un árbol y su definición. El nombre de la parte del árbol es la “clave” que queremos buscar y la descripción es el “valor”. Crearemos un “HashMap” formado por un texto para el nombre de la parte del árbol y otro para la definición del mismo (“<String, String>”, siendo el primero para la clave y el segundo para el valor; podríamos utilizar cualquier cosa como clave o como valor, como un número entero, un booleano, un objeto, etc). Con el método “put()” insertamos cada nombre de cada parte del árbol y su definición.

```
Map tablaHash = new HashMap<string, string>();

tablaHash.put("Raíz","Las raíces fijan el árbol al suelo. Las raíces pueden tener un...");
tablaHash.put("Tronco","El tronco sostiene la copa. Su capa exterior se llama corteza...");
tablaHash.put("Ramas","Las ramas suelen brotar a cierta altura del suelo...");
tablaHash.put("Hojas","A través de las hojas el árbol realiza la fotosíntesis y puede por lo
```

Ejercicio 1:

A. ¿Es una buena elección considerar para un método de hashing(trabajando con 210 claves) una función hash de la forma h(k)=k mod M con M=1+210? ¿Por qué?

Creo que no porque el hashing puede almacenar muchos datos pero hash no, no se muy bien es confuso 😕.

B. ¿Qué condición pondrías a N y M para que la elección de la función hash h(k)=(kxN)mod M 1<=k<=M fuese válida?¿Por qué?

No estoy segura pero creo que sería que N debería tener un valor menor a M.

Ejercicio 2:

¿Cómo podría utilizarse el hashing para implementar el TDA Array Disperso (array en el que la mayoría de sus elementos son nulos)?

Ejercicio 3:
Los empleados de una cierta compañía se representan en la base de datos por su nombre, número de empleado y número de la seguridad social.Construir una estructura de tablas hash que permita acceder al registro de un empleado por cualquiera de estos tres datos. (Nota:No se dispone de memoria suficiente para duplicar los registros de los empleados).

Si el ejemplo que vi arriba en la definición es correcto, entonces sería registrar los datos como:

- ***Map tablaHash = new HashMap<string, int, int>();***
- ***tablaHash.put("Alejandro", 234, núm seguridad);***

Y supongo que para acceder a los datos de un empleado debería pedir en pantalla un nombre, cuenta o número de seguridad o algo así; como el input que vimos en clase.

Ejercicio 4:
Se define el índice radial de una Tabla Hash abierta como el número de casillas de la tabla por el número de elementos de la lista enlazada con mayor número de los mismos presente en la tabla.Diseñar un algoritmo adecuado para calcular tal índice radial.

Índice Radial = Casillas x Elementos
Tal vez sea declarando a las casillas y los elementos con un tamaño fijo.