Participantes:

- **María Gallego Martín**
- **Yeray Granada Layos**
- **Daniel Rodríguez Manzanedo**
- **Carlos Gómez Robles**
- **Miguel Ángel López Roche**

Ejercicio para hacer en clase.

Dada la siguiente implementación del merge sort:

```
    public static void sort(int arr[], int l, int r)
    {
        if (l < r)
        {
            int m = (l+r)/2;

            sort(arr, l, m);
            sort(arr , m+1, r);

            merge(arr, l, m, r);
        }
    }

    public static void merge(int arr[], int l, int m, int r)
    {
        int n1 = m - l + 1;
        int n2 = r - m;

        int L[] = new int [n1];
        int R[] = new int [n2];

        for (int i=0; i<n1; ++i)
            L[i] = arr[l + i];
        for (int j=0; j<n2; ++j)
            R[j] = arr[m + 1+ j];
        int i = 0, j = 0;

        int k = l;
        while (i < n1 && j < n2)
        {
            if (L[i] <= R[j])
            {
                arr[k] = L[i];
                i++;
            }
            else
            {
```

```
                arr[k] = R[j];
                j++;
            }
            k++;
        }


    }
```

Al ejecutar el código obtenemos las siguientes salidas:

Input: [0, 1, 2, 3, 4] → Output: [0, 1, 2, 3, 4] (correcto)
Input: [4, 3, 2, 1, 0] → Output: [0, 0, 2, 0 ,0] (error)

Solución:

Lo primero que hemos hecho al crear la clase merge sort ha sido crear hacer un main para poder ejecutar las pruebas con los inputs: [0,1,2,3,4] y [4,3,2,1,0] y printear la salida. Al realizer la primera ejecución observamos como, tal cual dice el enunciado, el primer output si lo realiza bien pero el segundo no.

```java
public class MergeSort {
        public static void main(String [ ] args){

            int[] a1 = {0,1,2,3,4};
            int[] a2 = {4,3,2,1,0};

            sort(a1, 0 , a1.length-1);

            for (int i = 0; i < a1.length; i++) {
                System.out.println(a1[i]);
            }

            System.out.println("------------------------------------------------------------");

            sort(a2, 0 , a2.length-1);

            for (int j = 0; j < a2.length; j++) {
                System.out.println(a2[j]);
            }
        }
```

Lo siguiente ha sido comprobar si la función sort, tal cual viene dada, realiza un correcto funcionamiento. Tras las pruebas realizadas confirmamos que ejecuta la division de los array de forma correcta.

```java
public static void sort(int arr[], int l, int r)
{
    if (l < r)
    {
        int m = (l+r)/2;

        sort(arr, l, m);
        sort(arr , m+1, r);

        merge(arr, l, m, r);
    }
}
```

Por lo tanto, el error se encuentra en la función de merge. Tras hacer el debugging nos hemos dado cuenta que al tener en el while && cuando cumple una de las condiciones durante el merge sale del bucle, faltando así elementos del array por unir. Esto se ha solucionado mediante dos while que controlen que se copian los elementos restantes del lado derecho y del izquierdo.

Código dado:

```
public static void merge(int arr[], int l, int m, int r)
{
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[] = new int [n1];
    int R[] = new int [n2];

    for (int i=0; i<n1; ++i)
        L[i] = arr[l + i];
    for (int j=0; j<n2; ++j)
        R[j] = arr[m + 1+ j];
    int i = 0, j = 0;

    int k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
```

Lo añadido:

```
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }

}

}
```