LABORATORIO 2

EJERCICIO 1:

```
#include <bits/stdc++.h>

template <class T>
void print_vector(const std::vector<T>& v){
   for(const auto &i : v){
       std::cout << i << ' ';
   }
   std::cout << '\n';
}

template <class T>
void merge(std::vector<T>& v, int left_index, int mid_index, int
right_index){
   int i, j, k;
   int left_size = mid_index - left_index + 1;
   int right_size = right_index - mid_index;

   std::vector<T> left_vector(left_size), right_vector(right_size);
   for(i = 0; i < left_size; i++){
       left_vector[i] = v[left_index + i];
   }

   for(j = 0; j < right_size; j++){
       right_vector[j] = v[mid_index + 1 + j];
   }

   i = 0, j = 0, k = left_index;
   while((i<left_size) and (j<right_size)){
       if(left_vector[i] <= right_vector[j]){
           v[k] = left_vector[i];
           i++;
       }
       else{
           v[k] = right_vector[j];
           j++;
       }
       k++;
   }

   while(i < left_size){
       v[k] = left_vector[i];
       i++;
```

```
        k++;
    }

    while(i < right_size){
        v[k] = right_vector[j];
        j++;
        k++;
    }
}

template <class T>
void merge_sort(std::vector<T>& v, int left_index, int right_index){
    int mid_index;
    if(left_index < right_index){
        mid_index = (left_index + right_index) / 2;
        merge_sort(v, left_index, mid_index);
        merge_sort(v, mid_index + 1, right_index);

        merge(v, left_index, mid_index, right_index);
    }
}

int main(){
    std::vector<int> k = {7, 2, 14, 9, 24, 19, 38, 32, 56, 42};
    print_vector(k);
    merge_sort(k, 0, k.size()-1);
    print_vector(k);


    return 0;
}
```

EJERCICIO 2:

```
#include <bits/stdc++.h>
using namespace std;

int min(int x, int y){
    return (x < y) ? x : y;
}

void merge(vector<int>& A, vector<int>& temp, int from, int mid, int
to){
    int k = from, i = from, j = mid + 1;
```

```cpp
    while(i <= mid and j <= to){
        if(A[i] < A[j]){
            temp[k++] = A[i++];
        } else{
            temp[k++] = A[j++];
        }
    }

    while(i < A.size() and i <= mid){
        temp[k++] = A[i++];
    }

    for(int i = from; i <= to; i++){
        A[i] = temp[i];
    }
}

void mergeSort(vector<int>& A, vector<int>& temp, int low, int high){
    // dividir el array en bloques de size 'm'
    for(int m = 1; m <= high - low; m = 2*m){
        for(int i = low; i < high; i += 2*m){
            int from = i;
            int mid = i + m - 1;
            int to = min(i + 2*m - 1, high);

            merge(A, temp, from, mid, to);
        }
    }
}

void printArray(vector<int>& array){
    for(int i = 0; i < array.size(); i++){
        cout << array[i] << ' ';
    }
    cout << '\n';
}

int main(){
    int n = 10;
    vector<int> A(n), temp(n);

    for(int i = 0; i < n; i++){
        temp[i] = A[i] = rand() % 20;
    }

    cout << "Array Original: "  << '\n';
    printArray(A);

    mergeSort(A, temp, 0, n - 1);

    cout << "Array Ordenado: " << '\n';
    printArray(A);

    return 0;
}
```

EJERCICIO 3:

```
#include <bits/stdc++.h>
using namespace std;

void merge(int gArray[], int izq, int mid1, int mid2, int der, int
arr[]){
    int i = izq, j = mid1, k = mid2, l = izq;

    while ((i < mid1) && (j < mid2) && (k < der)){
        if(gArray[i] < gArray[j]){
            if(gArray[i] < gArray[k]) arr[l++] = gArray[i++];
            else
                arr[l++] = gArray[k++];
        } else{
            if(gArray[j] < gArray[k]) arr[l++] = gArray[j++];
            else arr[l++] = gArray[k++];
        }
    }

    while ((i < mid1) && (j < mid2)){
        if(gArray[i] < gArray[j]) arr[l++] = gArray[i++];
        else arr[l++] = gArray[j++];
    }

    while ((j < mid2) && (k < der)){
        if(gArray[j] < gArray[k]) arr[l++] = gArray[j++];
        else arr[l++] = gArray[k++];
    }

    while ((i < mid1) && (k < der)){
        if(gArray[i] < gArray[k]) arr[l++] = gArray[i++];
        else arr[l++] = gArray[k++];
    }
```

```
   while (i < mid1)
       arr[l++] = gArray[i++];

   while (j < mid2)
       arr[l++] = gArray[j++];

   while (k < der)
       arr[l++] = gArray[k++];
}


void mergeSort3Recursivo(int gArray[], int izq, int der, int arr[]){
   if (der - izq < 2)
       return;

   int mid1 = izq + ((der - izq) / 3);
   int mid2 = izq + 2 * ((der - izq) / 3) + 1;

   mergeSort3Recursivo(arr, izq, mid1, gArray);
   mergeSort3Recursivo(arr, mid1, mid2, gArray);
   mergeSort3Recursivo(arr, mid2, der, gArray);

   merge(arr, izq, mid1, mid2, der, gArray);
}

void mergeSort3(int gArray[], int n){
   if (n == 0)
       return;

   int fArray[n];

   for (int i = 0; i < n; i++)
       fArray[i] = gArray[i];

   mergeSort3Recursivo(fArray, 0, n, gArray);

   for (int i = 0; i < n; i++)
       gArray[i] = fArray[i];
}

int main(){
   int arr[] = {1, 7, 3, 0, 5, 8, 12, 98, 45};
   int size = sizeof(arr)/sizeof(arr[0]);

   mergeSort3(arr, size);
```

```
    for (int i = 0; i < size; i++){
        cout << arr[i] << ' ';
    }


    return 0;
}
```