MERGE SORT WEEK 12

```
C++ Assignments | Merge Sort | Week 12

1.Given an array of integers, sort it in descending order
using merge sort algorithm.
2.Reverse Pairs (Leetcode Problem) : Given an integer array
nums, return the number of reverse
pairs in the array.
A reverse pair is a pair (i, j) where:
0 <= i < j < nums.length and
nums[i] > 2 * nums[j].
```

1.

```
#include <iostream>
#include <vector>
using namespace std;

// Merge function to merge two halves of array in descending
order
void merge(vector<int>& arr, int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Create temporary arrays
    vector<int> L(n1), R(n2);

    // Copy data to temporary arrays L[] and R[]
    for (int i = 0; i < n1; ++i)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; ++j)
        R[j] = arr[mid + 1 + j];

    // Merge the temporary arrays back into arr: descending
order
    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] >= R[j]) {
            arr[k] = L[i];
            ++i;
        } else {
            arr[k] = R[j];
            ++j;
        }
        ++k;
    }

    // Copy the remaining elements of L[], if any
    while (i < n1) {
        arr[k] = L[i];
        ++i;
        ++k;
    }

    // Copy the remaining elements of R[], if any
    while (j < n2) {
        arr[k] = R[j];
        ++j;
        ++k;
    }
}

// Merge sort function to sort array in descending order
void mergeSort(vector<int>& arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        // Sort first and second halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        // Merge sorted halves
        merge(arr, left, mid, right);
    }
}

// Function to print array
void printArray(const vector<int>& arr) {
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;
}

int main() {
    vector<int> arr = {12, 11, 13, 5, 6, 7};
    int n = arr.size();

    cout << "Original array: ";
    printArray(arr);

    // Sort array in descending order using merge sort
    mergeSort(arr, 0, n - 1);

    cout << "Array sorted in descending order: ";
    printArray(arr);

    return 0;
}
```

2.

```
#include <iostream>
#include <vector>
using namespace std;

// Merge function to merge two sorted halves and count
reverse pairs
int mergeAndCount(vector<int>& nums, int left, int mid, int
right) {
    int count = 0;
    int j = mid + 1;

    for (int i = left; i <= mid; ++i) {
        while (j <= right && nums[i] > 2LL * nums[j]) {
            ++j;
        }
        count += (j - (mid + 1));
    }

    vector<int> temp(right - left + 1);
    int p1 = left, p2 = mid + 1;
    int p = 0;

    while (p1 <= mid && p2 <= right) {
        if (nums[p1] <= nums[p2]) {
            temp[p++] = nums[p1++];
        } else {
            temp[p++] = nums[p2++];
        }
    }

    while (p1 <= mid) {
        temp[p++] = nums[p1++];
    }
    while (p2 <= right) {
        temp[p++] = nums[p2++];
    }

    for (int k = 0; k < temp.size(); ++k) {
        nums[left + k] = temp[k];
    }

    return count;
}

// Merge sort function to sort array and count reverse pairs
int mergeSortAndCount(vector<int>& nums, int left, int right)
{
    if (left >= right) {
        return 0;
    }

    int mid = left + (right - left) / 2;
    int count = 0;

    count += mergeSortAndCount(nums, left, mid);
    count += mergeSortAndCount(nums, mid + 1, right);
    count += mergeAndCount(nums, left, mid, right);

    return count;
}

// Function to count reverse pairs in the array
int reversePairs(vector<int>& nums) {
    return mergeSortAndCount(nums, 0, nums.size() - 1);
}

int main() {
    vector<int> nums = {1, 3, 2, 3, 1};

    int count = reversePairs(nums);
    cout << "Number of reverse pairs: " << count << endl;

    return 0;
}
```