C++ Assignment Solutions | Merge Sort | Week 12

1. Given an array of integers, sort it in descending order using merge sort algorithm.

Code :

```
 #include <iostream>
#include <vector>

void merge(std::vector<int>& arr, int left, int middle, int
right) {
    int n1 = middle - left + 1;
    int n2 = right - middle;

    std::vector<int> L(n1), R(n2);

    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[middle + 1 + j];

    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++;
    }
```

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

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

void mergeSort(std::vector<int>& arr, int left, int right) {
    if (left < right) {
        int middle = left + (right - left) / 2;

        mergeSort(arr, left, middle);
        mergeSort(arr, middle + 1, ri
        merge(arr, left, middle, right);
    }
}

void mergeSortDescending(std::vector<int>& arr) {
    int size = arr.size();
    mergeSort(arr, 0, size - 1);
}

int main() {
    std::vector<int> arr = {12, 11, 13, 5, 6, 7};

    std::cout << "Original array: ";
    for (int num : arr) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    mergeSortDescending(arr);

    std::cout << "Sorted array in descending order: ";
    for (int num : arr) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

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].

   Code :

```
 class Solution {
private:
    void merge(vector<int>& nums, int low, int mid, int
high, int& reversePairsCount){
        int j = mid+1;
        for(int i=low; i<=mid; i++){
            while(j<=high && nums[i] > 2*(long long)nums[j])
{
                j++;
            }
            reversePairsCount += j-(mid+1);
        }
        int size = high-low+1;
        vector<int> temp(size, 0);
        int left = low, right = mid+1, k=0;
        while(left<=mid && right<=high){
            if(nums[left] < nums[right]){
                temp[k++] = nums[left++];
            }
            else{
                temp[k++] = nums[right++];
            }
        }
        while(left<=mid){
            temp[k++] = nums[left++];
        }
        while(right<=high){
            temp[k++] = nums[right++];
        }
        int m=0;
        for(int i=low; i<=high; i++){
            nums[i] = temp[m++];
        }
    }
```

```
    void mergeSort(vector<int>& nums, int low, int high,
int& reversePairsCount){
        if(low ⩾ high){
            return;
        }
        int mid = (low + high) >> 1;
        mergeSort(nums, low, mid, reversePairsCount);
        mergeSort(nums, mid+1, high, reversePairsCount);
        merge(nums, low, mid, high, reversePairsCount);
    }
public:
    int reversePairs(vector<int>& nums) {
        int reversePairsCount = 0;
        mergeSort(nums, 0, nums.size()-1,
reversePairsCount);
        return reversePairsCount;
    }
};
```

Note:- Please try to invest time doing the assignments which are necessary to build a strong foundation. Do not directly Copy Paste using Google or ChatGPT. Please use your brain.