```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

void mergeSortHelper(vector<int> *mainArray, int startIdx, int endIdx,
                     vector<int> *auxiliaryArray);
void doMerge(vector<int> *mainArray, int startIdx, int middleIdx, int endIdx,
             vector<int> *auxiliaryArray);

// Best: O(nlog(n)) time | O(n) space
// Average: O(nlog(n)) time | O(n) space
// Worst: O(nlog(n)) time | O(n) space
vector<int> mergeSort(vector<int> array) {
  if (array.size() <= 1) {
    return array;
  }
  vector<int> auxiliaryArray = array;
  mergeSortHelper(&array, 0, array.size() - 1, &auxiliaryArray);
  return array;
}

void mergeSortHelper(vector<int> *mainArray, int startIdx, int endIdx,
                     vector<int> *auxiliaryArray) {
  if (startIdx == endIdx) {
    return;
  }
  int middleIdx = (startIdx + endIdx) / 2;
  mergeSortHelper(auxiliaryArray, startIdx, middleIdx, mainArray);
  mergeSortHelper(auxiliaryArray, middleIdx + 1, endIdx, mainArray);
  doMerge(mainArray, startIdx, middleIdx, endIdx, auxiliaryArray);
}

void doMerge(vector<int> *mainArray, int startIdx, int middleIdx, int endIdx,
             vector<int> *auxiliaryArray) {
  int k = startIdx;
  int i = startIdx;
  int j = middleIdx + 1;
  while (i <= middleIdx && j <= endIdx) {
    if (auxiliaryArray->at(i) <= auxiliaryArray->at(j)) {
      mainArray->at(k++) = auxiliaryArray->at(i++);
    } else {
      mainArray->at(k++) = auxiliaryArray->at(j++);
    }
  }
  while (i <= middleIdx) {
    mainArray->at(k++) = auxiliaryArray->at(i++);
  }
  while (j <= endIdx) {
    mainArray->at(k++) = auxiliaryArray->at(j++);
  }
}

```