Write a c++ code for Merge sort

|  |
| --- |
| #include <iostream>  using namespace std;    // Merges two subarrays of array[].  // First subarray is arr[begin..mid]  // Second subarray is arr[mid+1..end]  void merge(int array[], int const left, int const mid,             int const right)  {      auto const subArrayOne = mid - left + 1;      auto const subArrayTwo = right - mid;        // Create temp arrays      auto \*leftArray = new int[subArrayOne],           \*rightArray = new int[subArrayTwo];        // Copy data to temp arrays leftArray[] and rightArray[]      for (auto i = 0; i < subArrayOne; i++)          leftArray[i] = array[left + i];      for (auto j = 0; j < subArrayTwo; j++)          rightArray[j] = array[mid + 1 + j];        auto indexOfSubArrayOne          = 0, // Initial index of first sub-array          indexOfSubArrayTwo          = 0; // Initial index of second sub-array      int indexOfMergedArray          = left; // Initial index of merged array        // Merge the temp arrays back into array[left..right]      while (indexOfSubArrayOne < subArrayOne             && indexOfSubArrayTwo < subArrayTwo) {          if (leftArray[indexOfSubArrayOne]              <= rightArray[indexOfSubArrayTwo]) {              array[indexOfMergedArray]                  = leftArray[indexOfSubArrayOne];              indexOfSubArrayOne++;          }          else {              array[indexOfMergedArray]                  = rightArray[indexOfSubArrayTwo];              indexOfSubArrayTwo++;          }          indexOfMergedArray++;      }      // Copy the remaining elements of      // left[], if there are any      while (indexOfSubArrayOne < subArrayOne) {          array[indexOfMergedArray]              = leftArray[indexOfSubArrayOne];          indexOfSubArrayOne++;          indexOfMergedArray++;      }      // Copy the remaining elements of      // right[], if there are any      while (indexOfSubArrayTwo < subArrayTwo) {          array[indexOfMergedArray]              = rightArray[indexOfSubArrayTwo];          indexOfSubArrayTwo++;          indexOfMergedArray++;      }      delete[] leftArray;      delete[] rightArray;  }    // begin is for left index and end is  // right index of the sub-array  // of arr to be sorted \*/  void mergeSort(int array[], int const begin, int const end)  {      if (begin >= end)          return; // Returns recursively        auto mid = begin + (end - begin) / 2;      mergeSort(array, begin, mid);      mergeSort(array, mid + 1, end);      merge(array, begin, mid, end);  }    // UTILITY FUNCTIONS  // Function to print an array  void printArray(int A[], int size)  {      for (auto i = 0; i < size; i++)          cout << A[i] << " ";  }    // Driver code  int main()  {      int arr[] = { 12, 11, 13, 5, 6, 7 };      auto arr\_size = sizeof(arr) / sizeof(arr[0]);        cout << "Given array is \n";      printArray(arr, arr\_size);        mergeSort(arr, 0, arr\_size - 1);        cout << "\nSorted array is \n";      printArray(arr, arr\_size);      return 0;  } |