#include <iostream>  
#include <chrono>  
  
using namespace std;  
  
void merge(int\* arr, int l, int mid, int r) {  
 int lSize, rSize;  
 lSize = mid - l + 1;  
 rSize = r - mid;  
  
 int \*arr\_left = new int[lSize + 1];  
 int \*arr\_right = new int[rSize + 1];  
  
 for (int i = l; i <= mid; i++) {  
 arr\_left[i - l] = arr[i];  
 }  
 for (int j = mid + 1; j <= r; j++) {  
 arr\_right[j - mid - 1] = arr[j];  
 }  
 arr\_left[lSize] = INT\_MAX;  
 arr\_right[rSize] = INT\_MAX;  
 int i = 0, j = 0;  
 for (int k = l; k <= r; k++) {  
 if (arr\_left[i] < arr\_right[j]) {  
 arr[k] = arr\_left[i];  
 i++;  
 } else {  
 arr[k] = arr\_right[j];  
 j++;  
 }  
 }  
 delete[] arr\_left;  
 delete[] arr\_right;  
}  
  
void mergeSort(int\* arr, int l, int r) {  
 if (l < r) {  
 int middle = (l + r) / 2;  
 mergeSort(arr, l, middle);  
 mergeSort(arr, middle + 1, r);  
 merge(arr, l, middle, r);  
 }  
}  
  
int main() {  
 using std::chrono::high\_resolution\_clock;  
 using std::chrono::duration\_cast;  
 using std::chrono::duration;  
 using std::chrono::milliseconds;  
  
  
 int len;  
 cout << "Enter the size of an array: ";  
 cin >> len;  
  
 auto start = high\_resolution\_clock::now();  
 int \*arr = new int[len];  
  
 for (int i = 0; i < len; i++) {  
 arr[i] = rand();  
 }  
  
 for(int i=0; i<len; i++) {  
 cout << arr[i] << " ";  
 }  
  
 mergeSort(arr, 0, len - 1);  
  
 auto end = high\_resolution\_clock::now();  
  
 cout<<endl;  
 for(int i=0; i<len; i++) {  
 cout << arr[i] << " ";  
 }  
  
 duration<double, std::milli> ms\_double = end - start;  
 cout <<endl<< "Execution time is: " << ms\_double.count() << "ms" << endl;  
 printf("%.2f", ms\_double.count() / 1000);  
 delete[]arr;  
 return 0;  
}

MergeSort() – to divide array till one-size element -> O(n);

Merge() – to compare elements and merge elements of array in ascending order -> O(log n);

Overall Time Complexity: O(n log n);

Worst, Average and Best Case - O(n\*log(n))

#include <iostream>  
#include <chrono>  
  
using namespace std;  
  
void merge(int\* arr, int l, int mid, int r) {  
 int lSize, rSize;  
 lSize = mid - l + 1;  
 rSize = r - mid;  
  
 int \*arr\_left = new int[lSize];  
 int \*arr\_right = new int[rSize];  
  
 for (int i = 0; i < lSize; ++i) {  
 arr\_left[i] = arr[l + i];  
 }  
 for (int j = 0; j < rSize; ++j) {  
 arr\_right[j] = arr[mid + 1 + j];  
 }  
  
/\* arr\_left[lSize] = INT\_MAX;  
 arr\_right[rSize] = INT\_MAX;\*/  
  
 int i = 0, j = 0;  
 for (int k = l; k <= r; k++) {  
 if(i==lSize)  
 arr[k] = arr\_right[j++];  
 else if( j == rSize)  
 arr[k] = arr\_left[i++];  
 else if(arr\_left[i] <= arr\_right[j])  
 arr[k] = arr\_left[i++];  
 else  
 arr[k] = arr\_right[j++];  
 }  
 delete[] arr\_left;  
 delete[] arr\_right;  
}  
  
void mergeSort(int\* arr, int l, int r) {  
 if (l < r) {  
 int middle = (l + r) / 2;  
 mergeSort(arr, l, middle);  
 mergeSort(arr, middle + 1, r);  
 merge(arr, l, middle, r);  
 }  
}  
  
int main() {  
 using std::chrono::high\_resolution\_clock;  
 using std::chrono::duration\_cast;  
 using std::chrono::duration;  
 using std::chrono::milliseconds;  
  
  
 int len;  
 cout << "Enter the size of an array: ";  
 cin >> len;  
  
 auto start = high\_resolution\_clock::now();  
 int \*arr = new int[len];  
  
 for (int i = 0; i < len; i++) {  
 cin>> arr[i];  
 }  
  
 for(int i=0; i<len; i++) {  
 cout << arr[i] << " ";  
 }  
  
 mergeSort(arr, 0, len - 1);  
  
 auto end = high\_resolution\_clock::now();  
  
 cout<<endl;  
 for(int i=0; i<len; i++) {  
 cout << arr[i] << " ";  
 }  
  
 duration<double, std::milli> ms\_double = end - start;  
 cout <<endl<< "Execution time is: " << ms\_double.count() << "ms" << endl;  
 printf("%.2f", ms\_double.count() / 1000);  
 delete[]arr;  
 return 0;  
}