#include <iostream>

#include <ctime>

#include <stdio.h>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Merge Sort \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int counter = 0;

void display(int \*arr, int size){

cout << "Sorted: "; /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (int i = 0; i<size; i++){ /\* A function to \*/

cout << arr[i] << " "; /\* display the array \*/

} /\* elements \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cout << endl;

cout<<"Counts through the whole sorting prosses: "<<counter;

cout << endl;

}

void merge(int \*arr, int \*arrLeft, int \*arrRight, int sizeLeft, int sizeRight, int sizeArray){

counter++;

int i = 0; // i to manage the index of the temporary left array

int j = 0; // j to manage the index of the temporary right array

int k = 0; // k to manage the index of the real array

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Sorting and Merging \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(i < sizeLeft && j < sizeRight) { // Loop untill the end of the Right array or the Left array is reached

counter++;

if (arrLeft[i] < arrRight[j] ) // If the number in the left array is less then the number in the right

arr[k++] = arrLeft[i++]; // put the number which was in the left array in the real array and increment both the real and the left arrays' indexes

else // If the number in the right array is less then the number in the left

arr[k++] = arrRight[j++]; // put the number which was in the right array in the real array and increment both the real and the right arrays' indexes

}

while (i < sizeLeft){ // If the Left array still contains elements

arr[k++] = arrLeft[i++]; // put them in the real array

counter++;

}

while (j < sizeRight){ // If the right array still contains elements

arr[k++] = arrRight[j++]; // put them in the real array

counter++;

}

}

void mergeSort(int \*arr, int size){

int \*arrLeft; // Initializing two temporary dynamic

int \*arrRight; // arrays to hold the two arrays to be merged

int sizeLeft; // The two arrays' sizes

int sizeRight;

counter++;

if (size > 1) { // checking if the size of the arrays divide bigger than 1

if (size % 2 == 0){ // if the size is even divide both arrays into 2 equal parts

sizeLeft = size / 2;

sizeRight = size / 2;

counter++;

}

else{ // else if the size is odd divide them so that the left array would have 1 more number than the right one

sizeLeft = (size / 2) + 1;

sizeRight = size / 2;

counter++;

}

arrLeft = new int[sizeLeft]; // Give the Left array its size

arrRight = new int[sizeRight]; // Give the Right array its size

/\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Dividing \*/

/\*\*\*\*\*\*\*\*\*\*\*\*/

for (int i = 0; i < sizeLeft; i++){ // Putting the first half of the numbers of the real array in the temporary left array

arrLeft[i] = arr[i];

counter++;

}

int j = 0; // j works as an index number for the temporary Right array

for (int i = sizeLeft; i < size; i++){ // Putting the second half of the numbers of the real array in the temporary right array

arrRight[j++] = arr[i];

counter++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Recursion \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*/

mergeSort(arrLeft, sizeLeft); // Recursively call the Merge Sort function for the Left array untill the size is less than or equals 1

mergeSort(arrRight, sizeRight); // Recursively call the Merge Sort function for the Right array untill the size is less than or equals 1

merge(arr, arrLeft, arrRight, sizeLeft, sizeRight, size); // Call the Merge function to sort and merge both halfs of the array

}

}

int main(){

int start = clock(); // Capturing the time in the beginning of the program

int \*Arr; // Initializing the dynamic array

int size; // An integer holding the dynamic array's size

cout << "Enter size of Array: ";

cin >> size; // Aquiring the size of the array from the user

cout << endl;

Arr = new int[size]; // Giving the dynamic array its size

for (int i = 0; i < size; i++){ // A loop to let the user enter Elements to be sorted

cout << "Enter number: ";

cin >> Arr[i];

}

cout << endl;

mergeSort(Arr, size); // Calling the Merge Sort Function

display(Arr, size); // A function for displaying the sorted numbers in the array

printf("Time Elapsed: %.2f s\n", (double)(clock() - start)/CLOCKS\_PER\_SEC); // Printing out the execution time of the program

// int stop\_s=clock();

// cout << "Execution time: " << (stop\_s-start\_s)/double(CLOCKS\_PER\_SEC)\*1000 << " m s"<<endl;

return 0;

}

#include<iostream>

using namespace std;

int counter = 0;

void display(int \*arr, int size){

cout << "Sorted: "; /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (int i = 0; i<size; i++){ /\* A function to \*/

cout << arr[i] << " "; /\* display the array \*/

} /\* elements \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cout << endl;

cout<<"Counts through the whole sorting prosses: "<<counter;

cout << endl;

}

void Swap(int &arr1,int &arr2){

int temp;

temp = arr1;

arr1 = arr2;

arr2 = temp;

counter++;

}

int partition(int \*arr, int firstIndex, int lastIndex){

int i;

i = firstIndex - 1;

int pivot = arr[lastIndex];

for (int j = firstIndex; j < lastIndex; j++){

if (arr[j] <= pivot){

i++;

Swap(arr[i], arr[j]);

}

}

Swap(arr[i + 1], arr[lastIndex]);

return i+1;

}

void quickSort(int \*arr, int firstIndex, int lastIndex){

if (firstIndex < lastIndex){

int p = partition(arr, firstIndex, lastIndex);

quickSort(arr, firstIndex, p - 1);

quickSort(arr, p + 1, lastIndex);

}

}

int main(){

int \*Arr;

int size;

cout << "Enter size of Array:";

cin >> size;

Arr = new int[size];

for (int i = 0; i < size; i++){

cout << "Enter number:";

cin >> Arr[i];

}

quickSort(Arr, 0,size-1);

display(Arr, size);

return 0;

}