```
/*
Yanes Garcia
This program finds the number of inversions of a list. It is an example of the Divide-and-
Conquer approach and Mergesort. W(n)= 2W(n/2) + n -1 for n>1, when n is a power of 2. For
example, the number of inversions of { 1, 2, 3, 4, 5 } is 0, and the number of inversions of { 1, 3,
2, 5, 4 } is 2.
*/

#include <iostream>
using namespace std;
int merge(int input[], int pointer[], int left, int mid, int right);
int mergeSort(int input[], int array_size);
int mergeSort1(int input[], int pointer[], int left, int right);
int main(){
	int input[] = { 1, 3, 2, 5, 4 };
	cout << "Number of inversions in a list are " << mergeSort( input, 5) << endl;
	system("pause");
	return 0;
}

int mergeSort(int input[], int size){
	int* pointer = new int[ sizeof(int) * size];
	return mergeSort1(input, pointer, 0, size - 1);
}

int mergeSort1(int input[], int pointer[], int left, int right){
	int mid;
	int swapp = 0;
	if (right > left) {
		mid = (right + left) / 2;
	swapp = mergeSort1(input, pointer, left, mid);
	swapp += mergeSort1(input, pointer, mid + 1, right);
	swapp += merge(input, pointer, mid + 1, left, right);
	}
	return swapp;
}

int merge(int input[], int pointer[], int mid, int left, int right) {
	int swapp = 0;
	int i = left;
	int j = mid;
	int k = left;
	while ((i <= mid - 1) && (j <= right)) {
		if (input[i] > input[j]) {
			pointer[k++] = input[j++];
```

```
                        swapp = swapp + (mid - i);
                } else {
                        pointer[k++] = input[i++];
                } }
        while (j <= right) {
                pointer[k++] = input[j++];
        }
        while (i <= mid - 1) {
                pointer[k++] = input[i++];
        }

        for (i = left; i <= right; i++) {
                input[i] = pointer[i];
        }
        return swapp;
}
```