Sorting by combining merge and insertion

Description We will combine two sorting algorithms, insertion-sort and merge-sort as follows: when the input size is less than 10, we will use insertion-sort; otherwise, we will use merge-sort. More specifically, we replace line 1 in Merge-Sort (page 34) with "if $r - p >= 10$", and add line 6 "else Insertion-Sort(A, p, r)". Insertion-Sort(A, p, r) implies performing insertion sort on the subarray $A[p \cdots r]$. In this lab assignment you will implement this hybrid sorting algorithm.

For testing your code, unzip "lab01.zip" to your working directory and go inside the lab01 folder. You will see several files. File "Grade01" is the file you will use to test your implementation. Your execution file must be named "MergeInsertion.exe". Perhaps you may need to use chmod to change the permissions of "Grade01". Towards this end, you can just type

"chmod 700 Grade01" or "chmod +x Grade01".

Now if you run "./Grade01" and your implementation is correct, you will see the following messages:

Correct for 1 th example.
Correct for 2 th example.
Correct for 3 th example.
Correct for 4 th example.
Correct for 5 th example.
Correct for 6 th example.
Correct for 7 th example.
Correct for 8 th example.
Correct for 9 th example.
Correct for 10 th example.

You can also find a summary of the execution in the file named "result".

You are required to explain and compile your code in front of the TA, and answer any questions you are asked about your code. The idea is that you should understand what your wrote. If you want to test your code for just one test file, you can try: ./MergeInsertion.exe < ../testfiles/t1

Finally, there is also a very simple "Makefile" for you, which is very convenient for compiling your file(s). If you type "make," it will compile MergeInsertion.cpp. If you have other files, you will need to modify the Makefile, there is plenty information on the web on how to do this. If you type "make clean", then MergeInsertion.exe will be deleted. You don't really have to use the provided "Makefile", but if you don't know how to use it, this is a good time to learn it. You can find abundant examples on the web.

Input structure The input starts with an integer number which indicates the number of elements (integers) to be sorted. Then, the elements follow, one per line.

Output structure Output the sorted sequence one element per line. Do *not* insert spaces at the beginning or at the end of any element.

Examples of input and output:

Input

```
6
3
6
23
76
4
56
```

Output

```
3
4
6
23
56
76
```

(If you are unsure how to get started take a look at the provided example .cpp file showing how to perform typical operations needed to read numbers from the input.)

Submission Before the posted deadline, submit a .zip or zipped tar archive of your program through the assignments page of CatCourses. Please use your UCMNetID as the filename for the zipped archive. Be careful since CatCourse strictly enforces the assignment deadlines. Note that submission alone is not enough, you must present your work to your TA before submission.

Important Note: Never change the files under the "testfiles" folder, or the grading script file Grade01. If you do so, it will be considered as SERIOUS CHEATING. Please read file AcademicHonestyPolicy.pdf available at CatCourses, and in particular note that the policy says: "if any violation of the UCM Academic Honesty Policy is suspected in a course, the instructor of record must fill out the Faculty Report for Academic Misconduct".