CSE 100 Algorithm Design and Analysis Lab 01 (20 points)
Due date: September 6, 23:59

Sorting by combining merge- and insertion-sort

Description We combine two sorting algorithms, insertion-sort and merge-sort as follows: when the input size is less than 100, we use insertion-sort; otherwise, we use merge-sort. More specifically, we replace line 1 in Merge-Sort (page 34) with "if $r - p >= 100$", 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. File "Grade01" is the file you will use to test your implementation. Your execution file must be named "MergeInsertion.exe". 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
```

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

test 1: success
test 2: success
test 3: success
test 4: success
test 5: success
test 6: success
test 7: success
test 8: success
test 9: success
test 10: success

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

If you want to test your code for just one test file, run:

./MergeInsertion.exe < ../testfiles/t1

There is also a simple "Makefile" which is very convenient for compiling your file(s). If you type "make", it will compile MergeInsertion.cpp. If you have other files, you need to modify "Makefile". There is plenty of information on the web on how to do this. If you type "make clean", then MergeInsertion.exe is deleted.

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.*

Example 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 Submit a "zip" or "tar.gz" archive of your program through the assignments page of CatCourses by the deadline. Use your UCMNetID as the file name. Be careful since CatCourse strictly enforces the assignment deadline. You may be asked to compile, run, and explain the code to the TA to prove that you understand what you wrote.

Grading We provide 10 test cases for you to try your implementation. Each of them is valued at 1 point if executed correctly. We will use 10 additional test cases to check that your implementation is general enough. These are not provided to you and are also valued at 1 point each.

Important Note We will use plagiarism software to detect cheating. Offenders will be subjected to the UCM Academic Honesty Policy which states: *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.*