CSE 100 Algorithm Design and Analysis Lab 02 (20 points)
Due date: September 20, 23:59

Sorting: Heap-Sort and Quick-Sort

Description In this assignment you will implement two different algorithms for sorting elements in an array: Heap-Sort and Quick-Sort. In Quick-Sort, use a *random* element of the array as pivot.

For testing your code, use "GradeQuick" and "GradeHeap" in your working directory. You also need to be sure the directory "testfiles" is in your working directory and that its content has not been modified.

For Heap-Sort, your execution file's name must be "HeapSort.exe". Likewise, for Quick-Sort, your execution file's name must be "QuickSort.exe". As in the previous lab, perhaps you will need to use chmod to change the execution permissions of "GradeQuick" and "GradeHeap". If so, you can just type:

```
chmod 700 GradeQuick
```

and

```
chmod 700 GradeHeap
```

If you run "./GradeQuick" 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".

Similarly, you will have to run "./GradeHeap" for testing Heap-Sort.

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

./QuickSort.exe < ../testfiles/t1

The included "Makefile" will compile both QuickSort.cpp and HeapSort.cpp. If you type "make clean", then QuickSort.exe and HeapSort.exe will be 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
```

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.*