

Due 1:30PM Wednesday, November 17

Name _____

Student ID _____

Instructions Follow instructions *carefully*, failure to do so will result in points being deducted. It is highly recommended that you write your homework using L^AT_EX or Word. Staple this cover page to the front of your assignment for easier grading. Be sure to submit a hardcopy of your problem answers and your program analysis in class. Clearly label each problem and submit the answers in order. Put your program analysis to the end. In addition, submit your homework including all the program files via the webhandin (<http://www.cse.unl.edu/~cse310/handin>). Late homework policy specified in course syllabus *will be strictly followed*. Be sure to show *sufficient work* to justify your answer(s). Numerical answers with no explanation will NOT be granted full points. If you are asked to prove something, you must give as formal, rigorous, and complete proof as possible. You are to work individually, and all work should be your own. The CSE academic dishonesty policy is in effect (http://cse.unl.edu/undergrads/academic_integrity.php).

Problem	Page	Points	Score
5.6.2	194	8	
5.6.3	194	10	
5.6.11	195	10	
6.3.2 (b)	222	6	
6.3.9	223	6	
6.4.5	230	20	
Program			
Correctness		20	
Style/Documentation		10	
Analysis		10	
Total		100	

Program

Implement two sorting algorithms: heapsort and insertion sort and investigate their performance on arrays of sizes $n = 10^2, 10^3, 10^4$ and 10^5 . For each of these sizes, consider

- a. randomly generated files of integers in the range $[1..n]$.
- b. increasing files of integers 1, 2, ..., n .
- c. decreasing files of integers $n, n-1, \dots, 1$.

The HeapSort.cpp and InsertionSort.cpp source files include the implementation of the two algorithms. In addition, you need to create a main.cpp file to do the empirical tests on your algorithms. To compare and contrast the two algorithms you will need to keep track of how many comparisons, swaps, and CPU time (see clock()) it takes for each instance to be sorted by each algorithm. For case a, you may want to run multiple randomly generated instances of n and take an average, the choice is yours.

To report the final results, the main program should output a nice looking table. The following is merely a suggestion. As long as it is readable and conveys all the necessary information you may design your own table.

Case a: Randomly Generated Files

n	10^2	10^3	...	10^5
HeapSort	comps: 32.5	comps: xx	...	comps: xx
	swaps: 4.3	swaps: xx	...	swaps: xx
	time: xx	time: xx	...	time: xx
InsertionSort	comps: 32.5	comps: xx	...	comps: xx
	swaps: 4.3	swaps: xx	...	swaps: xx
	time: xx	time: xx	...	time: xx

Case b: Increasing Files

n	10^2	10^3	...	10^5
HeapSort	comps: 32.5	comps: xx	...	comps: xx
	swaps: 4.3	swaps: xx	...	swaps: xx
	time: xx	time: xx	...	time: xx
InsertionSort	comps: 32.5	comps: xx	...	comps: xx
	swaps: 4.3	swaps: xx	...	swaps: xx
	time: xx	time: xx	...	time: xx

Case c: Decreasing Files

n	10^2	10^3	...	10^5
HeapSort	comps: 32.5	comps: xx	...	comps: xx
	swaps: 4.3	swaps: xx	...	swaps: xx
	time: xx	time: xx	...	time: xx
InsertionSort	comps: 32.5	comps: xx	...	comps: xx
	swaps: 4.3	swaps: xx	...	swaps: xx
	time: xx	time: xx	...	time: xx

Your program must be written in C++, execute correctly on the CSE unix environment, and compile using the g++ compiler using a makefile. The makefile should compile your program into an executable called SortTest.

Points will be awarded based on the following categories:

- **Correctness** – Your code should execute as described above. You must include a makefile that will compile your code into an executable called SortTest.
- **Style/Documentation** – Your code should be readable, properly indented and spaced with sufficient comments to explain. Good naming style should be followed.
- **Analysis** – Your analysis should reflect your understanding and critical analysis of the algorithms as well as demonstrate that you gave some thought to the assignment beyond mindlessly coding the algorithms. Analyze the algorithms and give time and space efficiencies as a conclusion. Discuss the advantages and disadvantages of each algorithms, expected behavior on various input configurations and how these expectations were in line with or at odds with your empirical findings. Discuss anything you think relevant, pitfalls that you faced, etc. Turn in your analysis as a hard copy with the rest of your assignment.