PONDER 11 : HEAP/MERGE SORTS

Due Saturday at 11:59 PM MST

The next programming assignment will be to implement the Heap Sort and an external Natural Merge Sort. The following web site might be useful in understanding how many of these sorts work: [www.sorting-algorithms.com](http://www.sorting-algorithms.com/).

DRIVER PROGRAM

Copy the templates for this assignment from "/home/cs235/week11\_E". Complete the driver program named week11.cpp that gets the "sort name" and the "file name" from the command line. If the sort name is "heap" you will read the file into a vector and sort the vector with your heap sort, then output the sorted file to the console. If the sort name is "merge" you will read the file into a linked list and sort the linked list with the merge sort.

Example command line execution:

week11 merge assign11\_test1.in

week11 heap assign11\_test2.in

The two test files to sort are located in the folder "/home/cs235/week11\_E"

There are also stub files for week11.cpp as well as heap.h and merge.h

Heap Sort

Create a function representing the Heap Sort algorithm. Please use the array implementation of a heap tree. Put your heap sort functions in a file name heap.h

Implement the heap sort as a function using the algorithm discussed in class. Use the STL vector container to store the data.

Your sort function should take a vector of items as input and sort the vector. Then output the sorted vector in your driver program. Use the heap sort algorithm to sort the vector.

Note: You must implement all the sort algorithm's. You my NOT use the STL's heap functions.  
  
You should implement the following functions:

* **heapSort(vector & toSort)**: Function you call to sort the passed vector
* **heapify()**: Turns a given vector into a heap.
* **percolateDown()**: This method takes an index as a parameter and fixes the heap from that index down.

As mentioned before, you need to use the array implementation of the heap. Pseudocode and descriptions of all these methods are presented in Chapter 13.2 of the textbook.

Merge Sort

Create a function template representing the Merge Sort. There are several variations to the merge sort. You must implement the external natural merge sort. Although a merge sort can be done with arrays, linked lists, or files. You are to do either a linked list or file implementation. You must follow a natural merge sort implementation. Place your merge sort functions in a file named merge.h   
  
Write the necessary function that implements the **natural merge sort** using linked lists or files. You must use the **external** natural merge sort algorithm discussed in class and that is in your text book.   
  
Use the STL list container to store the data but **DO NOT USE** the sort function or merge function from the list container!!!   
  
If you use a linked list implementation, your mergeSort function should take a single linked list as a parameter

Driver Program

Your driver program will be used to output the sorted data.

There is no test bed for this assignment. You simply need to output the sorted array or linked list to match the corresponding output files (assign11\_test1.out, assign11\_test2.out). Use "setw(3)" Compare your output with the two output files using the linux "diff" utility.

Submitting

Use the makefile provided to build your tar file.   
  
Your program will be graded according to the following rubric:

submit week11.tar

Your program will be graded according to the following rubric:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Exceptional 100% | Good  90% | Acceptable  70% | Developing  50% | Missing 0% |
| HeapSort  50% | The style and code quality are excellent, and the sort is perfectly implemented | The algorithm is accurately and efficiently implemented | The algorithm is correct and testBed runs without error | The essence of the algorithm is properly represented | No attempt was made |
| MergeSort 50% | The style and code quality are excellent, and the sort is perfectly implemented | The algorithm is accurately and efficiently implemented | The algorithm is correct and testBed runs without error | The essence of the algorithm is properly represented | No attempt was made |

Please make sure to fill out the program header in the makefile with the following information: the amount of coding time required by each to complete the assignment.