Csci 41: Introduction to Data Structures
Project 2

Directions: You must upload a single zip file (e.g., Myname-Proj2.zip) to Canvas. The file should contain the following:

a. All *.cpp and *h files
b. Screenshot(s) of the output/results by running your program.
c. Excel/Word file that summarizes and explains your execution results (10 points)
d. Within your source code, you should write down comments that describes the purpose, expected input, and expected output of each function. Inside each function, you may introduce comments to explain the logics of important lines of source code.

Please DO NOT submit exe file or the entire Visual Studio projects to me. All I need is your source code that ends with cpp and h.

All of the above items will be graded. Grading criteria include but not limited to the completeness and correctness of the source code, correctness of output, and sufficiency and understandability of your comments.

Honor Statement: By turning in your assignment to me, you confirm that all of the work is **your own**. That is, you did not take your answer and adapt it from someone else, or get detailed advice from another student on how they answered a question. You also cannot give away your answers to another student. You agree, by sending me the answers via email, that this is your own work. This also includes trying to find an answer on the web (if such answers exist, I will find all of them and make sure that your answer is unique). **Penalties for cheating and plagiarism range from a 0 or an F on a particular assignment, through an F for the course, to expulsion from the university.**

TurnItOn will be turned on that search all possible answers among classmates and external sources.

Objectives: Use the execution time recording source code learned from Section 1.4 (time.h) or (chrono) and compare the execution time of the following 5 implementations.

Introduce 1000000 (N) integers randomly and save them in a vector/array. Find the 100 (M) smallest elements using the following implementations. Note if your computer cannot handle 1 million elements, you may reduce the size. However, you are NOT allowed to reduce the size too small which result in insignificant execution time difference among all the algorithms. (You will get very low score if the execution time among 6 algorithms are insignificant).

1. Use shell sort and the h formula is A033622 (introduced by Sedgewick1986) from Wikipedia (https://en.wikipedia.org/wiki/Shellsort). Print the 100 smallest elements.
2. Use quick sort and then print the 100 smallest elements.
3. Use Dijkstra 3-way partition and then print the 100 smallest elements.
4. Implement a Priority Queue implemented by the following 3 different approaches. Print the 100 smallest element.
    a. Ordered array/vector
    b. Heap using array/vector
    c. (worth 60+ points) Heap using binary tree (a heap constructed by ***tree nodes***. If you linked list to construct a heap, you will get zero). You are required to implement the following functions *insert* and *delMax for priority queue, of course, also including swim and sink functions (again, node based.* Additionally, for the heap using binary tree, you are required to implement the following functions:
        i. [Easy] A computeHeight function for heap binary tree that will return the height of the tree.
        ii. [Easy] A computeLeaves function for heap binary tree will return the number of leaves of the tree.
        iii. [Easy] A *lookup*(int key) function for heap binary tree that will return true if key is found in the tree and return false otherwise.
        iv. [Mid] A *sameLevel*(Node* current) function that will return all the node pointers at the same level of current node (return a vector<Node*>*).
        v. [Mid] A *descendant*(Node* current, Node* aNode) function that return true if aNode is current node's descendant node. Otherwise, return false.
        vi. [Hard] A *isHeap* function for heap binary tree that will return true is all subtrees satisfy heap properties (root has max value in each subtree and complete tree). Return false otherwise.

For each of the above implementations, please record their execution times for 5 times and then average the results (6 different kinds to find 100 smallest elements from 1 billion elements). Compare the difference of execution times and self-explain your findings (this may be asked in your final exam). Note your finding should be in a Word or Excel format that presents the results of 6 algorithms, each is executed 5 times.