Heap Sort Practice Set

Basic Level

1. Sort the array `[4, 10, 3, 5, 1]` using heap sort and show the heap after each step.
2. True/False: Heap sort is an **in-place** sorting algorithm.
3. True/False: Heap sort is a **stable** sorting algorithm.
4. What is the worst-case time complexity of heap sort?
5. Convert the array `[7, 2, 6, 3, 9]` into a **max-heap**. Show each step.
6. Explain why we use a **max-heap** for ascending order sorting.
7. Identify the heap after one extraction of the max element from `[20, 18, 15, 13, 10, 12, 8]`.

Medium Level

8. Given the array `[12, 11, 13, 5, 6, 7]`, perform **heap sort** and show the heap after each extraction.
9. Write a **C++ function** to heapify a subtree rooted at index `i`.
10. Swap the root with the last element in a max-heap `[15, 10, 8, 5, 7]` and show the heap after re-heapifying.
11. Which of the following is **not required** in heap sort?
    a) Building heap
    b) Heapify
    c) Merge step

d) Swapping root with last element

12. Convert the array `[9, 4, 7, 1, -2, 6, 5]` into a **min-heap.**

13. Explain the difference between **building a heap** and **heapifying during extraction.**

Advanced Level

14. Implement **heap sort in C++ without using STL priority_queue.**

15. Sort `[9, 4, 7, 1, -2, 6, 5]` using heap sort and **draw the heap after each extraction.**

16. Explain why **heap sort is not stable**, giving an example.

17. Given a nearly sorted array `[2, 6, 3, 12, 56, 8]`, discuss whether heap sort or insertion sort is more efficient and why.

18. True/False: Heap sort requires extra space proportional to n.

19. Adapt heap sort to **sort a stream of data** where elements arrive one by one.

20. Given an array of **10^6 elements**, explain why heap sort is preferred over bubble sort or insertion sort in terms of **time complexity and space efficiency**