Name: _____ **Date:** _____ **Period:** _____

Sorting Algorithms Practice Worksheet

Instructions: Answer each question clearly and completely. Show your work where applicable. For multiple-choice questions, circle the letter of the best answer. For free-response and short-answer questions, write your answer in the space provided.

1. **Multiple Choice:** Which of the following sorting algorithms has the best average-case time complexity?

   (A) Bubble Sort

   (B) Insertion Sort

   (C) Merge Sort

   (D) Selection Sort

2. **Short Answer:** What is the worst-case time complexity of Quick Sort, and what condition causes it?

3. **Free Response:** Trace the execution of Merge Sort on the following array: $[38, 27, 43, 3, 9, 82, 10]$. Show each recursive split and the merge steps.

4. **Multiple Choice:** Which sorting algorithm works by repeatedly swapping adjacent elements if they are in the wrong order?

   (A) Merge Sort

   (B) Bubble Sort

   (C) Quick Sort

   (D) Heap Sort

5. **Short Answer:** Explain why Insertion Sort is efficient for nearly sorted data. What is its best-case time complexity?

6. **Free Response:** Perform one complete pass of Bubble Sort on the array $[5, 1, 4, 2, 8]$, showing the array after each swap. Then state whether the array is fully sorted after that pass.

7. **Multiple Choice:** Which of the following is **not** an in-place sorting algorithm?

   (A) Quick Sort

   (B) Heap Sort

   (C) Merge Sort

   (D) Insertion Sort

8. **Short Answer:** What is the key property of a heap that makes Heap Sort possible? Briefly describe how Heap Sort works.

9. **Free Response:** Given the array $[64, 34, 25, 12, 22, 11, 90]$, show the result after each pass of Selection Sort. How many passes are needed to fully sort the array?

10. **Multiple Choice:** Which sorting algorithm has a worst-case time complexity of $O(n^2)$ but an average-case of $O(n \log n)$?

    (A) Merge Sort

    (B) Quick Sort

    (C) Heap Sort

    (D) Bubble Sort

Answer Key

1. **C** (Merge Sort has average-case $O(n \log n)$; Bubble, Insertion, and Selection are $O(n^2)$.)

2. Worst-case time complexity is $O(n^2)$. This occurs when the pivot chosen is always the smallest or largest element (e.g., already sorted or reverse-sorted array with poor pivot selection).

3. **Trace:**
   Split: $[38, 27, 43, 3]$ and $[9, 82, 10]$
   Further splits until single elements:
   $[38, 27]$, $[43, 3]$, $[9, 82]$, $[10]$
   Merge $[38, 27] \rightarrow [27, 38]$; $[43, 3] \rightarrow [3, 43]$; $[9, 82] \rightarrow [9, 82]$
   Merge $[27, 38]$ and $[3, 43] \rightarrow [3, 27, 38, 43]$
   Merge $[9, 82]$ and $[10] \rightarrow [9, 10, 82]$
   Final merge: $[3, 27, 38, 43]$ and $[9, 10, 82] \rightarrow [3, 9, 10, 27, 38, 43, 82]$

4. **B** (Bubble Sort repeatedly swaps adjacent elements if they are out of order.)

5. Insertion Sort is efficient for nearly sorted data because it only makes a few comparisons and shifts; best-case time complexity is $O(n)$ (already sorted).

6. Pass 1:
   Compare 5 and 1: swap $\rightarrow [1, 5, 4, 2, 8]$
   Compare 5 and 4: swap $\rightarrow [1, 4, 5, 2, 8]$
   Compare 5 and 2: swap $\rightarrow [1, 4, 2, 5, 8]$
   Compare 5 and 8: no swap
   Array after pass: $[1, 4, 2, 5, 8]$. Not fully sorted (4 and 2 are out of order).

7. **C** (Merge Sort requires $O(n)$ extra space, so it is not in-place.)

8. A heap is a complete binary tree where each parent is greater (max-heap) or smaller (min-heap) than its children. Heap Sort builds a max-heap, then repeatedly swaps the root with the last element and reheapifies the reduced heap.

9. Pass 1: $[11, 34, 25, 12, 22, 64, 90]$
   Pass 2: $[11, 12, 25, 34, 22, 64, 90]$
   Pass 3: $[11, 12, 22, 34, 25, 64, 90]$
   Pass 4: $[11, 12, 22, 25, 34, 64, 90]$
   Pass 5: $[11, 12, 22, 25, 34, 64, 90]$ (no change)
   6 passes are needed (for $n = 7$, Selection Sort requires $n - 1 = 6$ passes).

10. **B** (Quick Sort has worst-case $O(n^2)$ but average-case $O(n \log n)$.)