Z5007: Worked Solutions & Midterm Review Guide

Programming and Data Structures (Weeks 1–8) · Quiz 1 Practice Set (Q1–Q40) + AVL, Heap & Hash Bridge · innocent@iitmz.ac.in

This guide pairs step-by-step worked answers for all 40 questions in `Z5007_Quiz1_Practice40.pdf` (Weeks 1–6) with a compact Weeks 7–8 review covering AVL rotations, binary heaps, and hash tables. Work through each prompt independently before reading the derivation below.

Section A: Multiple Choice Worked Answers (Q1–Q15)

Q1. **(A)** A queue specifies FIFO behavior independently of its underlying circular array or linked-list storage, making it an abstract data type (ADT).

Q2. **(B)** Infinite loops violate **finiteness**, which requires termination after finitely many steps on every valid input.

Q3. **(D)** n^3 grows strictly faster than n^2, so $n^3 \notin O(n^2)$.

Q4. **(B)** $n \log_2 n$ dominates $2n$ asymptotically, yielding $f(n) = \Theta(n \log n)$.

Q5. **(C)** At $n = 1024$: $n^2/2 = 524{,}288$, exceeding $100n = 102{,}400$, $5000 \log_2 n = 50{,}000$, and $n \log_2 n = 10{,}240$.

Q6. **(B)** Binary search cuts the search interval in half each comparison: $\lceil \log_2(1000 + 1) \rceil = 10$ comparisons at most.

Q7. **(A)** Doubling capacity copies fewer than $2n$ elements across n appends, giving $O(1)$ amortized cost per append.

Q8. **(A)** On already sorted input, each of the $n - 1$ elements $i = 1 \ldots n - 1$ makes exactly 1 comparison with $a[i - 1]$ and stops, totaling $n - 1$ comparisons.

Q9. **(B)** Parentheses force addition first: $(3 + 4)$ becomes `3 4 +`, then multiplying by 5 yields `3 4 + 5 *`.

Q10. **(A)** Push 1, 2, 3 (stack $[1, 2, 3]$); pop → **3**; push 4 (stack $[1, 2, 4]$); pop → **4**; pop → **2**. Pop order: **3, 4, 2**.

Q11. **(C)** Master Theorem with $a = 2, b = 2, d = 0$: since $d = 0 < \log_2 2 = 1$ (Case 1), $T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$.

Q12. **(B)** First-come first-served ordering matches a FIFO **queue**.

Q13. **(B)** Four dequeues advance `head` by 4 modulo 8: $(5 + 4) \bmod 8 = \mathbf{1}$.

Q14. **(C)** Without a tail-predecessor link, reaching the second-to-last node requires scanning from `head` in $O(n)$ time.

Q15. **(B)** Choosing a uniformly random pivot gives expected balanced partitions and expected $O(n \log n)$ runtime on every input.

Section B: Code Tracing & Implementation Answers (Q16–Q30)

Q16. **Bubble Sort on** $[9, 4, 7, 3]$**:** Pass 1 swaps $(9, 4)$, $(9, 7)$, $(9, 3)$ → $[4, 7, 3, 9]$ (3 swaps). Pass 2 swaps $(7, 3)$ → $[4, 3, 7, 9]$ (1 swap). Pass 3 swaps $(4, 3)$ → $[3, 4, 7, 9]$ (1 swap). Total = **5 swaps**, equal to the exact number of inversions in $[9, 4, 7, 3]$: $(9, 4), (9, 7), (9, 3), (4, 3), (7, 3)$.

Q17. **Selection Sort on** $[64, 25, 12, 22, 11]$**:** Pass 1 swaps 64 and 11 → $[11, 25, 12, 22, 64]$. Pass 2 swaps 25 and 12 → $[11, 12, 25, 22, 64]$. Unstable because long-distance swaps can move an earlier duplicate key past a later equal key (for example, $[2_a, 2_b, 1] \to [1, 2_b, 2_a]$).

Q18. **Insertion Sort blanks:** (1) `j >= 0 and a[j] > key`; (2) `a[j + 1] = a[j]`; (3) `a[j + 1] = key`.

Q19. **Binary Search trace for** 23 **in** $[2, 5, 8, 12, 16, 23, 38, 56, 72, 91]$**:** Step 1: `lo=0, hi=9, mid=4, a[4]=16 < 23` → `lo=5`. Step 2: `lo=5, hi=9, mid=7, a[7]=56 > 23` → `hi=6`. Step 3: `lo=5, hi=6, mid=5, a[5]=23 == 23` (found at index 5).

Q20. **Linear vs. Binary Search** ($n = 10^6$)**:** (a) Linear search checks up to 1,000,000 items; (b) Binary search checks at most $\lceil \log_2(10^6 + 1) \rceil = 20$ items. Binary search requires sorted order so discarding half the array after one comparison never discards the target key.

Q21. **Merge stability:** With `<=`, $(2, a)$ precedes $(2, b)$ (stable). Switching to `<` picks from `right` on ties, producing $[(2, b), (2, a)]$ (unstable).

Q22. **Merge cost** ($m = 5, n = 3$)**:** Smallest = 3 comparisons when all items of the length-3 list precede the length-5 list (e.g. $[1, 2, 3]$ and $[4, 5, 6, 7, 8]$). Largest = $m + n - 1 = 7$ comparisons when lists interleave up to the final item (e.g. $[1, 3, 5, 7, 9]$ and $[2, 4, 8]$).

Q23. **Lomuto Partition on** $[8, 3, 6, 2, 7, 5]$ **(pivot** 5**):** Scans move 3 and 2 forward: swaps $(8, 3)$, $(8, 2)$, and final pivot swap $(6, 5)$ (note indices: $[3, 2, 5, 8, 7, 6]$), pivot ends at index **2**, total swaps = **3**.

Q24. Quickselect from Q23: Target index 2 matches pivot index 2 directly, so **5** is already the 3rd smallest (no further recursion). For the 5th smallest (index $4 > 2$), recurse into the right subarray `[8, 7, 6]`.

Q25. Bracket check on `([)]`: Push `(` → ['(']; push `[` → ['(', '[']; see `)`, pop `'['` $\neq$ `'('` → mismatch, immediately return **False**.

Q26. Circular queue bug: Without `% cap`, `tail` advances past index 3 (reaching index 4, `IndexError` on the 5th enqueue after 2 dequeues). Fix: update both pointers modulo capacity via `tail = (tail + 1) % cap` and `head = (head + 1) % cap`.

Q27. Reverse linked list blanks: (1) `curr.next = prev`; (2) `prev = curr`; (3) `curr = nxt`. Result for $1 \to 2 \to 3 \to 4$: $4 \to 3 \to 2 \to 1 \to$ None.

Q28. Linked-list `dequeue` blanks: (1) `self.head.val`; (2) `self.head.next`; (3) `self.tail = None` (prevents dangling tail reference when last node leaves).

Q29. Recursion `h(13)`: Calls `h(6)`, `h(3)`, `h(1)`, `h(0)`, then prints remainders on unwind: `1101`, which converts positive integer n to its **binary representation**.

Q30. Tower of Hanoi (5 discs, move 20): Move number k moves disc $d = \nu_2(k) + 1$ (1 plus the count of trailing zero bits in k). Since $20 = 4 \times 5 = 2^2 \times 5$ ($\nu_2(20) = 2$), move 20 moves **disc 3**.

Section C: Analysis, Proofs & Design Answers (Q31–Q40)

Q31. Big-O proof: For all $n \geq 1$, $5n^2 \leq 5n^3$ and $1 \leq n^3$, so $2n^3 + 5n^2 + 1 \leq 8n^3$. Choosing $c = 8$ and $n_0 = 1$ satisfies the formal definition $f(n) = O(n^3)$.

Q32. Loop 1: Outer loop multiplies i by 3, running $\lceil \log_3 n \rceil$ iterations; each runs an inner $O(n)$ loop. Tightest bound: $O(n \log n)$.

Q33. Loop 2: First nested block costs $10n = \Theta(n)$; second nested block costs $n^2 = \Theta(n^2)$. Total is $\Theta(n+n^2) = \mathbf{\Theta(n^2)}$, dominated by the second loop pair.

Q34. Choosing a sort: (a) **Insertion sort** (low code/RAM overhead, fast for $n = 12$); (b) **Insertion sort** (runs in $O(n)$ on nearly sorted input); (c) **Merge sort** (strict $O(n \log n)$ worst-case guarantee when extra RAM is available).

Q35. Master Theorem ($T(n) = 8T(n/2) + n^3$): $a = 8, b = 2, d = 3$, so $\log_b a = \log_2 8 = 3 = d$ (Case 2). Tight bound: $\Theta(n^3 \log n)$.

Q36. Recurrences: (a) Binary search: $T(n) = T(n/2) + O(1) \Rightarrow \mathbf{\Theta(\log n)}$ (Case 2 with $a = 1, b = 2, d = 0$); (b) Merge sort: $T(n) = 2T(n/2) + \Theta(n) \Rightarrow \mathbf{\Theta(n \log n)}$ (Case 2 with $a = 2, b = 2, d = 1$).

Q37. Unrolling ($T(n) = T(n-1) + 3, T(0) = 2$): $T(n) = T(0) + 3n = 3n + 2 = \mathbf{\Theta(n)}$.

Q38. Hanoi induction: Base case $n = 1$: $2^1 - 1 = 1 = M(1)$. Inductive step: assume $M(k) = 2^k - 1$; then $M(k+1) = 2(2^k - 1) + 1 = 2^{k+1} - 1$. Thus $M(8) = 2^8 - 1 = \mathbf{255}$.

Q39. Data structure selection: (a) **Stack** ($O(1)$ push/pop for undo history); (b) **Doubly linked list** ($O(1)$ splice insert/delete given node pointer); (c) **Min-heap** ($O(\log n)$ insert and extract-min for earliest deadline).

Q40. Quicksort worst case on $[1 \ldots 10]$: Selecting the last item on sorted input produces partitions of size $n - 1$ and 0 at every step, making $\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2} = \frac{10 \times 9}{2} = \mathbf{45}$ comparisons. Choosing a uniformly random pivot destroys adversarial sorted order and restores $O(n \log n)$ expected comparisons.

Midterm Bridge Review: Weeks 7–8 (AVL Trees, Heaps & Hash Tables)

1. AVL Balance Factor & Four Rotation Cases

For every node u, $bf(u) = h(u.\text{left}) - h(u.\text{right}) \in \{-1, 0, +1\}$, bounding height by $h < 1.44 \log_2(n+2)$. After insertion or deletion, rebalance the lowest ancestor z with $|bf(z)| = 2$: **LL** ($bf(z) = +2, bf(z.\text{left}) \geq 0$) → single right rotation at z; **RR** ($bf(z) = -2, bf(z.\text{right}) \leq 0$) → single left rotation at z; **LR** ($bf(z) = +2, bf(z.\text{left}) = -1$) → left rotate child then right rotate z; **RL** ($bf(z) = -2, bf(z.\text{right}) = +1$) → right rotate child then left rotate z.

2. Binary Max-Heap Build ($O(n)$) vs. Hash Table Collision Resolution

Bottom-up Floyd `heapify` sifts nodes from index $\lfloor n/2 \rfloor - 1$ down to 0; summing $\sum_{h=0}^{\lfloor \log n \rfloor} \frac{n}{2^{h+1}} h = O(n)$ yields linear-time construction, followed by n extractions in $O(n \log n)$ for in-place Heapsort. In hash tables with load factor $\alpha = n/m$: **Separate Chaining** yields $O(1+\alpha)$ expected lookup and clean deletion, whereas **Linear Probing** uses contiguous cache-friendly slots at index $(h(k) + i) \bmod m$ and requires tombstone sentinels so deletions never truncate probe chains.