Psiog Digital – Practice Test Solutions

Section 1: Aptitude & Logical Reasoning - Solutions

1. Work problem (A in 5 days, B in 10 days; A starts for 2 days then B joins):
A's rate = 1/5 per day, B's rate = 1/10 per day. In 2 days A does 2*(1/5)=2/5 of the work. Remaining = 1-2/5=3/5. Combined rate = 1/5+1/10=3/10 per day. Time to finish = (3/5) / (3/10) = (3/5)*(10/3) = 2 days. So they need 2 more days.

2. Train speed (150m train crosses 300m tunnel in 40.5s):
Distance covered = train length + tunnel length = 150 + 300 = 450 meters. Time = 40.5 seconds. Speed (m/s) = 450 / 40.5 = 11.111... m/s. Convert to km/h: *3.6 → 11.111... * 3.6 = 40 km/h.

3. Rectangle area after changes (length +20%, breadth -20%, new area = 192):
Let original length = L, breadth = B. New length = 1.2L, new breadth = 0.8B. New area = 1.2*0.8*L*B = 0.96*original_area = 192. So original_area = 192 / 0.96 = 200 sq m.

4. Odd one out: 3, 5, 11, 14, 17, 21
Look for patterns: parity, primes, differences. Primes in list: 3,5,11,17 are primes; 14 and 21 are composite. 14 and 21 are both composite; which single odd? 21 is divisible by 3 and 7; 14 by 2 and 7. If asked single odd, 11 differs because it's the only two-digit prime? Better reasoning: differences: +2, +6, +3, +3, +4 — no clean pattern. The clearest: 14 is only even number, so 14 is the odd one out.

5. Probability: drawing two cards, one Ace and one King:
Number of ways to draw 2 cards from 52 = C(52,2) = 1326. Ways to draw one Ace (4 choices) and one King (4 choices) = 4*4 = 16. So probability = 16/1326 = 8/663 ≈ 0.01206.

Section 2: Coding / Data Structures & Algorithms - Solutions

Two Sum (Hash map) — O(n) average time

```
def two_sum(nums, target):
    seen = {}
    for i, val in enumerate(nums):
        need = target - val
        if need in seen:
            return [seen[need], i]
        seen[val] = i
    return None
```

Merge Two Sorted Arrays (in-place for arrays with extra space or classic merge)

```
def merge_sorted(a, b):
    i = j = 0
    res = []
    while i < len(a) and j < len(b):
        if a[i] <= b[j]:
            res.append(a[i]); i += 1
        else:
            res.append(b[j]); j += 1
    res.extend(a[i:]); res.extend(b[j:])
    return res
```

Reverse a singly linked list (iterative)

```
class Node:
    def __init__(self, val, nxt=None):
        self.val = val; self.next = nxt

def reverse(head):
```

```
    prev = None
    curr = head
    while curr:
        nxt = curr.next
        curr.next = prev
        prev = curr
        curr = nxt
    return prev  # new head
```

Detect Cycle in Linked List (Floyd's Tortoise and Hare)

```
def has_cycle(head):
    slow = fast = head
    while fast and fast.next:
        slow = slow.next
        fast = fast.next.next
        if slow is fast:
            return True
    return False
```

Longest Substring Without Repeating Characters (sliding window)

```
def length_of_longest_substring(s):
    start = 0
    seen = {}
    max_len = 0
    for i, ch in enumerate(s):
        if ch in seen and seen[ch] >= start:
            start = seen[ch] + 1
        seen[ch] = i
        max_len = max(max_len, i - start + 1)
    return max_len
```

Valid Parentheses (stack)

```
def is_valid(s):
    pairs = {'(':')','[':']','{':'}'}
    stack = []
    for ch in s:
        if ch in pairs:
            stack.append(ch)
        else:
            if not stack or pairs[stack.pop()] != ch:
                return False
    return not stack
```

Maximum Subarray (Kadane's algorithm)

```
def max_subarray(nums):
    best = curr = nums[0]
    for x in nums[1:]:
        curr = max(x, curr + x)
        best = max(best, curr)
    return best
```

Section 3: System Design & Use-Case - Sample Approaches

1. Quiz Website - design summary:
Data model: questions(id, text, choices, answer, topic, difficulty, tags, media_url). API: GET /api/quiz?n=10&topics;=...; POST /api/answer. Strategy: use random sampling with exclusion of recent question_ids stored in user's profile or session. For large pools use reservoir sampling or random index + fetch by id. Cache hot topic sets; precompute random buckets. To avoid repetition across sessions keep a per-user history or use probabilistic decay (lower chance to repeat). Consider pagination, rate-limits, authentication, and CDN for media.

2. Ride-hailing cancellations - detecting fraud:
Collect features: cancellation timestamps, cancellation reasons, user history (rate of cancellations), driver ratings, geolocation patterns (multiple cancellations clustered), time between booking and cancellation, device fingerprint, payment attempts. Use rule-based signals (e.g., >X cancellations in Y days) plus a supervised model trained on labeled past fraud cases. Add human review for flagged accounts, soft penalties (temporary blocks), and appeals process. Monitor false positives and adjust thresholds.

3. URL Shortener - notes:
Use short code generation (base62) mapped to original URL in a database. APIs: POST /shorten, GET / -> redirect. Use cache (Redis) for hot lookups, consistent hashing to shard databases, and background jobs for analytics. Consider rate-limiting, custom aliases, and collision handling. For scale use distributed ID generation (Snowflake-like) or encode sequential IDs in base62 with obfuscation.

4. Elevator scheduling - approach:
Define objective (minimize average wait or total travel). Simple solution: nearest-car dispatch, with direction-aware pickup. More advanced: model as optimization / multi-agent RL or use heuristics like look-ahead where elevator picks requests that minimize added waiting time. Consider grouping stops (elevator stops along the way) and peak-time strategies (express to high floors).

Section 4: Behavioral / HR - Sample Answers & Tips

1. Challenging project (STAR format example):
Situation: Migration of monolith to microservices under tight deadline. Task: Split user-auth module and ensure zero downtime. Action: Identified bounded contexts, created API contracts, introduced feature flags, wrote integration tests, and gradually rolled out. Result: Migration completed with zero customer-facing downtime and 15% faster deploys.

2. Team conflict example:
Describe briefly: present the conflict, focus on listening to both sides, propose a compromise backed by data, and agree on measurable checkpoints. Emphasize communication and learning.

3. Where in 5 years:
Show growth mindset—e.g., 'I aim to be a senior engineer contributing to architecture decisions, mentoring juniors, and driving product-quality improvements.'

4. Why Psiog Digital:
Tailor this to your motivations: mission, tech stack alignment, opportunities to grow, or specific projects/products the company has worked on.

5. Solving with minimal guidance:
Describe a quick example where you researched, prototyped, sought minimal reviews, and iterated. Highlight initiative and outcomes.