Contents

Chapter 3 — Intuitive MCQ Bank (Prompt Engineering)

56 multiple-choice questions, intuitive/conceptual level. One correct option each. Cover the answer (blockquote) while testing yourself. Bilingual explanations (English + বাংলা).

Exam style: "Which statement best describes…", exactly one correct option; use full technical terms, no abbreviations.

বাংলা ব্যাখ্যা: এই ব্যাংকে অধ্যায় ৩-এর ৫০+ ধারণামূলক MCQ আছে প্রতিটি প্রশ্নের নিচে উত্তর ও সংক্ষিপ্ত ব্যাখ্যা (ইংরেজি + বাংলা)

Topic: Introduction and the Engineering Principle (Section 3.0)

Q1. A team finds that a foundation model handles a new task poorly. Which course of action best reflects the lecture's engineering principle?

- A. Immediately fine-tune the model on task-specific data before trying anything else.
- B. First push the prompt as far as it can go, and fine-tune only if a gap remains.
- C. Retrain the model from scratch so its weights match the task.
- D. Abandon the model, because prompting cannot change behavior without weight changes.

> **Answer: B.** The principle is "first maximize what prompts can achieve, then fine-tune if a gap remains," because prompting is fast, cheap, and reversible. A and C jump to heavier, slower methods prematurely; D is false — prompting *is* the technique that changes behavior without touching weights. **বাংলা:** আগে প্রম্পট, পরে fine-tuning — প্রম্পট দ্রুত ও reversible বলে এটাই নীতি

Q2. Why does the lecture call prompt engineering "high-leverage"?

- A. Because each prompt permanently rewrites a portion of the model's weights.
- B. Because prompts require expensive hardware to run.
- C. Because small changes in the input wording can cause large changes in model behavior.
- D. Because prompting can only be done by machine-learning experts.

> **Answer: C.** "High-leverage" means small wording changes produce large behavioral changes (slide 220). A is false — prompting never modifies weights; B and D contradict the lecture's claim that prompting is cheap and accessible. **বাংলা:** ছোট শব্দ-পরিবর্তনে বড় আচরণ-পরিবর্তন হয় বলেই "high-leverage"

Q3. Which statement best describes the relationship between prompt engineering and weight modification?

- A. Prompt engineering performs a single gradient step on the weights.
- B. Prompt engineering and fine-tuning both update weights, only at different rates.
- C. Prompt engineering freezes the input but updates the weights.
- D. Prompt engineering adapts model behavior without modifying weights.

> **Answer: D.** Prompting changes the *input*, leaving weights frozen — its defining property and the reason it is reversible. A, B, and C all wrongly involve weight updates, which belong to fine-tuning (Chapter 6). **বাংলা:** প্রম্পটিং ওজন বদলায় না, শুধু ইনপুট সাজায়; তাই এটি reversible

Q4. A reason the lecture highlights prompting as "reversible" is that:

- A. changing the prompt leaves the underlying model untouched, so you can revert instantly by changing the text back.
- B. you can always undo a gradient update with another gradient update.
- C. the model stores every previous prompt and can roll back automatically.
- D. reversibility is guaranteed by constrained decoding.

Answer: A. Because weights are never altered, switching prompts is instantly reversible — just change the text. B describes fine-tuning mechanics; C invents non-existent storage; D confuses reversibility with output-format enforcement. **বাংলা:** ওজন অটুট থাকে বলে টেক্সট বদলালেই আগের অবস্থায় ফেরা যায়

Topic: System, User, and Assistant Prompts (Section 3.1)

Q5. Which statement best describes the role hierarchy in modern chat-based large language models?

- A. The assistant history dominates because recency bias gives it the highest priority.
- B. The user prompt always overrides the system prompt because the user issues the task.
- C. The system prompt has the highest priority, then the user prompt, then the assistant history.
- D. All three roles carry equal weight; only token position matters.

Answer: C. Priority order is system > user > assistant history (slide 222). A confuses recency *bias* with *priority*; B inverts the hierarchy — system rules override user instructions; D ignores that roles are explicitly trained, not merely positional. **বাংলা:** অগ্রাধিকার: system > user > assistant — recency bias আর priority এক জিনিস নয়

Q6. A developer wants a hard safety rule that the user cannot talk the model out of. Where should the rule go, and why?

- A. In a user message, because user messages have the highest priority.
- B. In the assistant history, because recency bias makes it strongest.
- C. Anywhere, since all roles enforce rules equally.
- D. In the system prompt, because it is trained to override conflicting user instructions.

Answer: D. Safety rules belong in the system prompt: the hierarchy installed during alignment trains system instructions to override conflicting user instructions, so the rule persists. A and C misstate the hierarchy; B confuses influence-on-style with rule enforcement. **বাংলা:** কঠিন নিয়ম system prompt-এ রাখো — alignment-এ এটিকে user-এর ওপরে অগ্রাধিকার দিতে শেখানো হয়েছে

Q7. Why does the assistant history exert a strong influence on the next generated token even though it sits lowest in the priority hierarchy?

- A. Because of recency bias — content near the end of the context most strongly shapes the next token.
- B. Because it is special-tokenized differently from the other roles.
- C. Because the assistant role overrides the system prompt in long sessions.
- D. Because the model retrains on its own outputs after each turn.

Answer: A. The history is influential due to recency bias (it sits near the end of the context), yet it still cannot override system rules. B is irrelevant to influence strength; C contradicts the hierarchy; D is false — no retraining occurs. **বাংলা:** assistant ইতিহাস শেষে থাকায় recency bias-এ প্রভাবশালী, তবু system নিয়ম ভাঙতে পারে না

Q8. The role hierarchy (system > user > assistant) is best described as:

- A. a fixed property of the transformer architecture.
- B. a side effect of the tokenizer's special tokens.
- C. a behavior learned during post-training and alignment.
- D. an artifact of the sampling temperature.

Answer: C. The roles and their priority are introduced during post-training / alignment (Chapter 2.10) — learned behavior, not architecture. A is the classic distractor (it is *not* architectural); B and D confuse mechanism with formatting and sampling. **বাংলা:** ভূমিকা-অগ্রাধিকার আর্কিটেকচারে নেই — alignment-এ শেখানো আচরণ

Q9. A developer manually concatenates role strings into one text blob instead of using the model's chat template. What is the most likely consequence?

- A. Behavior degrades, because chat models expect roles wrapped in model-specific special tokens.

- B. Nothing changes, since chat templates are purely cosmetic.
- C. The model behaves more reliably because raw text is simpler.
- D. The system prompt automatically gains even higher priority.

Answer: A. Chat models special-tokenize roles (for example `<|system|>`, `<|user|>`); skipping the chat template breaks the learned role behavior. B and C understate the importance of the template; D invents a non-existent effect. **বাংলা:** chat টেমপ্লেট ছাড়া ভূমিকা special token-এ মোড়ানো হয় না, তাই আচরণ ভেঙে যায়

Q10. A medical-assistant system prompt says "provide general guidance only." A user asks "What disease do I have?" Which outcome best reflects the role hierarchy?

- A. The model gives a confident specific diagnosis because the user explicitly asked.
- B. The model ignores the system prompt because the user request is more recent.
- C. The model deletes the system prompt and answers freely.
- D. The model refuses to diagnose, gives general causes, and urges professional care, constrained by the system prompt.

Answer: D. The system prompt (highest priority) constrains the answer to general guidance, so the model declines a specific diagnosis. A and B wrongly let the user override the system rule; C is impossible — the model cannot delete its system prompt. **বাংলা:** system prompt সর্বোচ্চ অগ্রাধিকার বলে মডেল নির্দিষ্ট রোগনির্ণয় এড়িয়ে সাধারণ পরামর্শ দেয়

Topic: Zero-Shot Prompting (Section 3.2)

Q11. Which statement best describes zero-shot prompting?

- A. The prompt provides an instruction plus several worked examples of the task.
- B. The model is fine-tuned on zero examples before being deployed.
- C. The prompt provides only an instruction and no examples; the model relies on pretrained knowledge and alignment.
- D. The model produces an answer without receiving any prompt at all.

Answer: C. Zero-shot means instruction only, leaning on pretraining and alignment (slide 223). A describes few-shot; B confuses prompting with fine-tuning; D is meaningless — there is always a prompt. **বাংলা:** Zero-shot = শুধু নির্দেশ, উদাহরণ নেই; মডেল pretrain ও alignment-এর ওপর ভরসা করে

Q12. Why is zero-shot prompting often used as the *first* experiment before investing in few-shot or advanced prompting?

- A. Because it is fast, simple, low-token, and a strong baseline thanks to emergent capabilities.
- B. Because it guarantees the highest possible accuracy on every task.
- C. Because it permanently improves the model's weights for later tasks.
- D. Because it is the only technique that works on classification.

Answer: A. Zero-shot is cheap, fast, and a good baseline; you escalate only if it fails. B overstates it — zero-shot fails on specialized or ambiguous tasks; C involves weight changes (false); D is wrong — many techniques handle classification. **বাংলা:** Zero-shot দ্রুত, সস্তা ও ভালো baseline; ব্যর্থ হলে তবেই few-shot-এ যাও

Q13. A task is highly specialized and ambiguous, and the zero-shot output is inconsistent. According to the lecture, what is the most appropriate next step?

- A. Conclude the model is fundamentally incapable and stop.
- B. Lower the temperature to zero and accept the result unchanged.
- C. Repeat the identical zero-shot prompt many times until it works.
- D. Add clarification, constraints, or examples — for instance move to few-shot prompting.

Answer: D. The lecture says zero-shot may need clarification, constraints, or examples; few-shot fills exactly this gap. A gives up too early; B addresses determinism, not the ambiguity; C ignores

that the prompt itself is the limitation. **বাংলা:** Zero-shot ব্যর্থ হলে নির্দেশ স্পষ্ট করো বা উদাহরণ যোগ করো — অর্থাৎ few-shot-এ যাও

Q14. Which scenario is the *best* candidate for sticking with zero-shot prompting rather than escalating?

- A. A simple summarization task where latency matters and a quick baseline suffices.
- B. A highly specialized extraction task with a strict, unusual output format.
- C. A multi-step reasoning problem where the model keeps making arithmetic slips.
- D. A task whose correct output format the model never produces without examples.

Answer: A. Zero-shot suits simple tasks where speed matters and it is a reasonable baseline. B and D need examples (few-shot) for format control; C calls for chain-of-thought. **বাংলা:** সহজ কাজ + দ্রুততা দরকার = zero-shot যথেষ্ট; কঠিন ফরম্যাট/যুক্তিতে অন্য কৌশল লাগে

Topic: Few-Shot Prompting (Section 3.3)

Q15. Which statement best describes *why* few-shot prompting works, mathematically?

- A. The demonstrations trigger a gradient update that adjusts the weights slightly.
- B. The demonstrations change the conditioning context while the parameters stay frozen.
- C. The demonstrations are memorized into a vector database for later retrieval.
- D. The demonstrations raise the sampling temperature so the model explores more.

Answer: B. Few-shot conditions the same frozen model on extra (input, output) pairs — this is in-context learning, no weight update (Section 4.1). A and C invent learning/storage that does not occur; D confuses examples with sampling settings. **বাংলা:** উদাহরণ শুধু conditioning বদলায়, ওজন frozen থাকে — এটাই in-context learning

Q16. A developer adds three labeled examples to a prompt and accuracy improves. Which mechanism best explains this, per the lecture?

- A. The model fine-tuned itself on the three examples during inference.
- B. The examples increased the model's context window size.
- C. In-context learning let the model infer the input-output pattern and lock the format.
- D. The examples reduced the number of parameters the model uses.

Answer: C. Transformers infer the input-output relationship from demonstrations, clarifying intent and forcing the correct format/style. A wrongly claims weight updates; B and D misdescribe what examples do — they fill context, not resize or shrink the model. **বাংলা:** উদাহরণ থেকে মডেল প্যাটার্ন ও ফরম্যাট ধরে — in-context learning; কোনো ওজন বদলায় না

Q17. The lecture warns that "poor examples cause poor performance." Which scenario best illustrates this trap?

- A. Three clean, diverse, correctly labeled examples in a consistent format.
- B. A single representative example that matches the target distribution.
- C. Examples placed right before the query to exploit recency.
- D. Several noisy, inconsistent, or mislabeled examples that amplify the wrong pattern.

Answer: D. Noisy or mislabeled examples teach the wrong mapping and can make few-shot worse than zero-shot. A and B are *good* practice; C is a recommended placement, not a failure mode. **বাংলা:** এলোমেলো/ভুল-লেবেল উদাহরণ ভুল প্যাটার্ন শেখায় — তাই ফল খারাপ হয়

Q18. Why might adding *more* few-shot examples sometimes hurt rather than help?

- A. Because examples consume context space and can push other relevant content out of the window, and noisy ones amplify bias.
- B. Because each example silently overwrites a layer of the model's weights.
- C. Because the model is only ever allowed to read one example at a time.
- D. Because more examples always force the temperature to zero.

Answer: A. Examples cost context tokens and can crowd out relevant material; poor ones amplify bias (one clean example can beat five noisy ones). B is false (no weight changes); C and D are fabricated constraints. **বাংলা:** উদাহরণ কনটেক্সট খায় ও দরকারি তথ্য সরিয়ে দিতে পারে; খারাপ উদাহরণ পক্ষপাত বাড়ায়

Q19. A risk specific to few-shot prompting is that the model may:

- A. refuse to answer because examples confuse the system prompt.
- B. permanently learn the examples and apply them to unrelated future sessions.
- C. copy a specific example verbatim instead of generalizing the pattern.
- D. ignore the user prompt entirely whenever examples are present.

Answer: C. The lecture lists "the model might copy specific examples instead of generalizing" as a limitation. A and D overstate the effect; B is false — nothing persists across sessions without weight updates. **বাংলা:** মডেল প্যাটার্ন সাধারণীকরণ না করে নির্দিষ্ট উদাহরণ হুবহু নকল করতে পারে

Q20. The lecture says (a) put the cleanest/most relevant examples *first* and (b) place few-shot examples right *before* the query. How should these be reconciled?

- A. They contradict each other, so one must be ignored.
- B. Always put every example in the exact middle of the context.
- C. Order is irrelevant because transformers weigh all positions equally.
- D. Anchor the task with strong examples early, and keep query-relevant material late — primacy and recency both help.

Answer: D. "Anchor early, query-relevant material late" exploits both primacy and recency bias. A misreads them as contradictory; B lands content in the dead zone; C denies position bias outright. **বাংলা:** শুরুতে শক্ত উদাহরণে নোঙর, প্রশ্ন-সংশ্লিষ্ট অংশ শেষে — primacy ও recency দুটোই কাজে লাগে

Q21. Which statement best captures the difference between zero-shot and one-shot prompting?

- A. One-shot uses one demonstration of the task; zero-shot uses none.
- B. One-shot fine-tunes on one example; zero-shot fine-tunes on none.
- C. One-shot uses one model; zero-shot uses zero models.
- D. One-shot raises temperature once; zero-shot keeps it at zero.

Answer: A. One-shot = exactly one worked example (k = 1); zero-shot = no examples (k = 0). B wrongly invokes fine-tuning; C and D are nonsensical reinterpretations of "shot." **বাংলা:** One-shot = একটি উদাহরণ (k=1), zero-shot = কোনো উদাহরণ নেই (k=0)

Topic: Chain-of-Thought Prompting (Section 3.4)

Q22. Which statement best describes what happens internally during chain-of-thought prompting?

- A. The model runs an internal planner that executes each subtask in a separate inference pass.
- B. The model produces intermediate reasoning steps token by token, autoregressively, in a single inference run.
- C. The model calls an external calculator for each intermediate step.
- D. The model retrieves complete reasoning chains from a database.

Answer: B. Chain-of-thought is one inference run, generated token by token, with no internal loop, planner, or multi-pass process (slide 229). A and C describe agent/tool architectures (Chapter 5); D describes retrieval. **বাংলা:** একটাই inference run, ধাপগুলো autoregressive টোকেনে তৈরি — ভেতরে লুপ/প্ল্যানার নেই

Q23. Why does writing out intermediate steps help the model on a multi-step arithmetic problem?

- A. Because each emitted step enters the context and conditions the next tokens, splitting one hard prediction into several easy ones.
- B. Because the steps trigger a hidden gradient update mid-generation.
- C. Because the steps are computed by a separate internal arithmetic unit.
- D. Because writing steps lowers the model's parameter count.

Answer: A. Each generated step becomes context for the following tokens, decomposing one hard prediction into easy local ones. B and C invent mechanisms the lecture explicitly denies; D is meaningless. **বাংলা:** প্রতিটি ধাপ কনটেক্সটে যোগ হয়ে পরের ধাপকে সহজ করে — এক কঠিন কাজ অনেক সহজ কাজে ভাগ হয়

Q24. The strong benefit of chain-of-thought appearing only in large models is cited as evidence for:

- A. constrained decoding.
- B. recency bias.
- C. emergent abilities.
- D. prompt injection.

Answer: C. Wei et al. 2022 found the effect appears only in large models — evidence for emergent abilities. A, B, and D are unrelated concepts from other sections. **বাংলা:** বড় মডেলেই সুফল দেখা যায় — এটি emergent ability-র প্রমাণ

Q25. Which ordering reflects the lecture's finding about chain-of-thought variants?

- A. Zero-shot chain-of-thought significantly outperforms few-shot chain-of-thought.
- B. Both perform identically on every task.
- C. Neither variant improves over plain zero-shot prompting.
- D. Few-shot chain-of-thought significantly outperforms zero-shot chain-of-thought.

Answer: D. The lecture states few-shot chain-of-thought significantly outperforms zero-shot chain-of-thought. A inverts this; B and C contradict the reported accuracy gains. **বাংলা:** few-shot chain-of-thought, zero-shot chain-of-thought-এর চেয়ে স্পষ্টভাবে ভালো

Q26. Why can chain-of-thought prompting actually *hurt* on a simple single-step recall task?

- A. Because the generated chain is itself text that may introduce a wrong intermediate step the model then conditions on, plus it adds tokens and latency.
- B. Because the model cannot generate text on simple tasks.
- C. Because single-step tasks require an external planner that chain-of-thought disables.
- D. Because chain-of-thought forces the temperature above zero.

Answer: A. On tasks the model can do directly, the chain is extra generated text that can introduce errors (which it then conditions on) and costs tokens/latency. B is false; C and D invent unrelated constraints. **বাংলা:** সহজ কাজে অতিরিক্ত যুক্তি-টেক্সট ভুল ধাপ ঢোকাতে পারে এবং টোকেন/বিলম্ব বাড়ায়

Q27. Which statement best describes self-consistency chain-of-thought?

- A. Generate one chain at temperature zero and trust it fully.
- B. Generate several chains and always pick the longest one.
- C. Generate several chains at temperature above zero and take the majority answer.
- D. Generate one chain and average its digits.

Answer: C. Self-consistency samples K chains at temperature greater than zero and votes by majority (Section 4.2). A produces identical chains (degenerate vote); B and D are not the voting rule. **বাংলা:** একাধিক চেইন ($T > 0$) স্যাম্পল করে সংখ্যাগরিষ্ঠ উত্তর নেওয়া

Q28. Why must self-consistency use a temperature greater than zero?

- A. Because temperature zero makes the model refuse to answer.
- B. Because at temperature zero all chains are identical, so majority voting reduces to one sample and gains nothing.
- C. Because higher temperature guarantees every chain is correct.
- D. Because temperature zero disables the system prompt.

Answer: B. Diversity across chains is the mechanism; at temperature zero decoding is deterministic, all chains coincide, and voting is degenerate. A and D are false; C overstates — higher temperature does not guarantee correctness. **বাংলা:** $T = 0$ হলে সব চেইন একরকম, ভোট অর্থহীন; বৈচিত্র্যই ভোটকে কার্যকর করে

Q29. Five chains vote: three say "6", one says "5", one says "8". Two chains were wrong, yet the final answer is correct. What does this illustrate?

- A. Voting always requires all chains to agree.
- B. The minority answer is selected to break ties.
- C. Self-consistency ignores the extracted answers and picks randomly.
- D. Voting can recover the correct answer because wrong chains tend to scatter across different wrong answers while correct ones agree.

Answer: D. Correct reasoning converges on one answer; errors scatter, so the majority recovers the truth even when some chains fail. A, B, and C all misstate majority voting. **বাংলা:** সঠিক যুক্তি একই উত্তরে মেলে, ভুলগুলো ছড়িয়ে পড়ে — তাই সংখ্যাগরিষ্ঠ ভোটে সঠিকটা জেতে

Q30. Self-consistency lifts a 0.70-accurate reasoner to about 0.84 at K = 5, but practical K stays around 5 to 40. Why not use K = 1000?

- A. Because cost grows linearly in K while accuracy gains saturate — diminishing returns.
- B. Because more than 40 chains always reduces accuracy below the single-chain value.
- C. Because the majority vote becomes invalid past 40 chains.
- D. Because temperature must drop to zero for large K.

Answer: A. Cost is K inference runs (linear), but the vote's accuracy saturates once $p > 0.50$, so extra chains buy little. B and C invent false breakdowns; D contradicts the temperature requirement. **বাংলা:** খরচ K-এর সাথে রৈখিক বাড়ে, কিন্তু নির্ভুলতা থিতিয়ে যায় — তাই বেশি K অপচয়

Topic: Context Management and Lost in the Middle (Section 3.5)

Q31. Which statement best describes "lost in the middle"?

- A. Information in the middle of a long context is used worst, producing a U-shaped accuracy curve.
- B. Information at the very start of a long context is used worst.
- C. All positions in the context are used equally well.
- D. Information at the very end of a long context is used worst.

Answer: A. Accuracy is U-shaped: high at the start (primacy) and end (recency), lowest in the middle (Liu et al. 2023). B and D name the *high*-influence positions; C denies position bias. **বাংলা:** মাঝের তথ্য সবচেয়ে কম ব্যবহৃত হয় — accuracy U-আকৃতির

Q32. In the 20-document study, the relevant document in the middle gives roughly 54 percent accuracy while the closed-book baseline is about 56 percent. What is the striking implication?

- A. Placing the key document in the middle can be worse than giving no document at all.
- B. The middle is the optimal place for important documents.
- C. Closed-book answering is always more accurate than any retrieval.
- D. Position has no measurable effect on accuracy.

Answer: A. Middle placement (~54%) falls *below* the no-document baseline (~56%) — the strongest argument for careful ordering. B is the opposite of the finding; C overgeneralizes one comparison; D contradicts the U-curve. **বাংলা:** মাঝে রাখলে ডকুমেন্ট না দেওয়ার চেয়েও খারাপ ফল হতে পারে

Q33. Why does repeating a format requirement at the *end* of a long prompt improve reliability?

- A. Because the end of the context is the dead zone for attention.
- B. Because repetition increases the context window size.
- C. Because later text is special-tokenized as a system prompt.
- D. Because the end sits in the high-influence recency region closest to the generated tokens.

Answer: D. Instruction reiteration places the rule in the recency region, the strongest influence on the next tokens (technique 4). A mislabels the end as the dead zone (that is the *middle*); B and C are fabricated effects. **বাংলা:** শেষে নিয়ম পুনরাবৃত্তি করলে তা recency অঞ্চলে পড়ে — পরের টোকেনে সবচেয়ে প্রভাবশালী

Q34. A long legal contract must be analyzed. Which context-management technique does the lecture recommend, and why?

- A. Chunk it into coherent, labeled sections (for example "Chunk 2: Liability Clause") and feed only relevant chunks.
- B. Stuff the entire contract into one block so no information is lost.
- C. Place the most important clause in the exact middle of the context.
- D. Convert the contract into examples and rely on recency alone.

Answer: A. Chunking splits long documents into coherent, titled sections, avoiding mid-concept splits and feeding only relevant chunks — cleaner inputs, fewer hallucinations. B invites lost-in-the-middle; C targets the dead zone; D misuses the document. **বাংলা:** লম্বা ডকুমেন্ট অর্থপূর্ণ, নামকরণ-করা অংশে ভাগ করো এবং কেবল প্রাসঙ্গিক অংশ দাও

Q35. A travel-planning chat has run for many turns and the history is bloating the context. Which technique best preserves long-term state without exceeding the window?

- A. Message pruning of every constraint the user mentioned.
- B. Instruction reiteration of every past greeting.
- C. Summarization that compresses the history while preserving constraints, decisions, and preferences (for example "5-day Japan trip, 2000 euro budget").
- D. Formatting anchors applied to deleted turns.

Answer: C. Summarization replaces long histories with a task-focused summary that keeps constraints and decisions — maintaining state within the window. A would *delete* the constraints you must keep; B and D misapply techniques to irrelevant content. **বাংলা:** সারসংক্ষেপ করে ইতিহাস সংকুচিত করো অথচ শর্ত/সিদ্ধান্ত ধরে রাখো

Q36. What is the purpose of message pruning?

- A. To add more greetings and acknowledgments for politeness.
- B. To move the user's task into the middle of the context.
- C. To duplicate the system prompt at every turn.
- D. To remove irrelevant turns (greetings, outdated details) so the context carries less noise and generation is more stable.

Answer: D. Pruning drops irrelevant turns and outdated content, reducing noise for more stable, predictable generation. A adds the noise pruning removes; B and C describe other (or harmful) actions. **বাংলা:** অপ্রাসঙ্গিক টার্ন (অভিবাদন, পুরোনো তথ্য) ছেঁটে কোলাহল কমাও — আউটপুট স্থিতিশীল হয়

Q37. The lecture's recommended ordering ends with the user's immediate task placed *last*. Why?

- A. Because the last position is the dead zone, hiding the task from the model.
- B. Because the task must be special-tokenized as a system prompt.
- C. Because ordering has no effect on transformer attention.
- D. Because the last position is in the recency region, so the immediate task strongly drives the next tokens.

Answer: D. Putting the immediate task last keeps it in the high-influence recency region. A mislabels the end as the dead zone; B confuses roles; C denies position bias. **বাংলা:** ব্যবহারকারীর প্রশ্ন শেষে রাখলে তা recency অঞ্চলে পড়ে — পরের টোকেনে প্রবল প্রভাব ফেলে

Q38. Which statement best captures why context management matters at all for transformers?

- A. Transformers ignore most of the context, so its contents are irrelevant.
- B. Transformers attend to all tokens, so irrelevant or noisy content reduces accuracy and important information can get lost.
- C. Transformers process only the system prompt and discard everything else.
- D. Transformers weigh the middle of the context most strongly.

Answer: B. Because the model attends to *all* tokens, noise hurts and key information can be lost — hence the need to curate the context. A and C understate what the model reads; D inverts the position bias (middle is weakest). **বাংলা:** মডেল সব টোকেনে মনোযোগ দেয়, তাই কোলাহল নির্ভুলতা কমায় ও দরকারি তথ্য হারায়

Q39. A formatting-anchor technique is best described as:

- A. deleting all structure so the model is free to format however it likes.
- B. placing the schema only in the middle of the context.
- C. using consistent output templates (JSON, tables, bullet structures), with examples and repeated schemas when reliability matters.
- D. raising the temperature to encourage format variety.

Answer: C. Formatting anchors use consistent templates, examples, and repeated schemas for stable, predictable output. A removes the very structure the technique provides; B targets the dead zone; D works against format stability. **বাংলা:** সামঞ্জস্যপূর্ণ টেমপ্লেট (JSON/টেবিল) ও দরকারে স্কিমা পুনরাবৃত্তি দিয়ে আউটপুট ফরম্যাট স্থির রাখা

Topic: Prompt Patterns (Section 3.6)

Q40. A user asks an ambiguous, poorly specified question. Which prompt pattern most directly addresses this before solving?

- A. Hard constraints (strict rules).
- B. Rewriting the problem ("Rewrite the task in your own words. Then solve it.").
- C. Style control.
- D. Soft steering.

Answer: B. Rewriting the problem restates an ambiguous query in simpler terms before solving — exactly its stated use case. A enforces format, C adjusts tone, D nudges behavior — none resolves ambiguity directly. **বাংলা:** অস্পষ্ট প্রশ্নে "rewriting the problem" — আগে নিজের ভাষায় লিখে তারপর সমাধান

Q41. Which pattern best fits a safety-sensitive medical assistant that must avoid unsafe outputs?

- A. Tree-of-thought exploration.
- B. Program-like decomposition.
- C. Guardrails through prompting (style + constraints to prevent unsafe outputs).
- D. Style control alone.

Answer: C. Guardrails through prompting combine style and constraints to prevent unsafe outputs and connect to alignment (Chapter 2.10). A and B target reasoning/algorithmic tasks; D alone does not enforce safety. **বাংলা:** নিরাপত্তা-সংবেদনশীল কাজে "guardrails through prompting" — স্টাইল ও বাধা মিলে অনিরাপদ আউটপুট ঠেকায়

Q42. Which statement best distinguishes *hard constraints* from *soft steering*?

- A. Hard constraints raise temperature; soft steering lowers it.
- B. Hard constraints apply only to the assistant role; soft steering only to the system role.
- C. They are identical; the names are interchangeable.
- D. Hard constraints are explicit rules that must be followed; soft steering influences behavior without strict rules.

Answer: D. Hard constraints are enforceable rules ("Respond using valid JSON"); soft steering is flexible guidance ("Be thorough but concise"). A confuses them with sampling; B invents a role split; C denies a real distinction. **বাংলা:** Hard constraint = অবশ্য-পালনীয় নিয়ম; soft steering = কঠোর নিয়ম ছাড়াই দিকনির্দেশ

Q43. A team needs an analysis that draws on both legal and financial expertise. Which pattern fits best?

- A. Style control.
- B. Role decomposition (analyze as a legal expert, then a financial expert, then combine).
- C. Hard constraints.
- D. Rewriting the problem.

Answer: B. Role decomposition adopts several expert roles and merges their outputs — built for multi-domain reasoning. A adjusts tone; C enforces format; D handles ambiguity. **বাংলা:** বহু-ক্ষেত্র বিশ্লেষণে "role decomposition" — একাধিক বিশেষজ্ঞ-ভূমিকা নিয়ে ফল একত্র করা

Q44. Which statement best describes how real production prompts use these patterns?

- A. Exactly one pattern may be used per prompt; mixing is forbidden.
- B. Patterns replace the need for a system prompt entirely.
- C. Patterns only apply to image models, not text.
- D. Patterns are typically combined — for example persona + hard constraints + few-shot examples + instruction reiteration.

Answer: D. The lecture says real prompts combine several patterns (persona + constraints + few-shot + reiteration is the standard stack). A forbids the combining the lecture recommends; B and C are unrelated and false. **বাংলা:** বাস্তব প্রম্পটে কয়েকটি প্যাটার্ন একসাথে মেলানো হয় — persona + constraint + few-shot + reiteration

Q45. The critique-and-refine pattern is best characterized as:

- A. asking the model to call an external tool before answering.
- B. asking the model to branch into multiple strategies and pick one.
- C. asking the model to restate the problem in simpler words.
- D. asking the model to critique a draft first, then produce an improved version.

Answer: D. Critique-and-refine evaluates the draft for clarity/correctness, then improves it — used for quality, writing, and code review. A is tool use; B is tree-of-thought; C is rewriting the problem. **বাংলা:** খসড়াকে আগে সমালোচনা, পরে উন্নত সংস্করণ তৈরি — critique-and-refine

---

Topic: Structured Outputs and Function Calling (Section 3.7)

Q46. Which statement best describes the goal of structured outputs?

- A. To make the model produce longer, more creative free-form prose.
- B. To turn the model into a deterministic component whose output is predictable and machine-readable.
- C. To remove the system prompt from the pipeline.
- D. To guarantee the factual truth of every field.

Answer: B. Structured outputs make models behave like deterministic components in software, producing machine-readable output for reliable integration. A is the opposite goal; C is unrelated; D overstates — structure controls format, not truth. **বাংলা:** আউটপুটকে যন্ত্র-পাঠযোগ্য ও নির্ধারিত করা যাতে সফটওয়্যারে নির্ভরযোগ্যভাবে যুক্ত হয়

Q47. Constrained decoding makes JSON output valid by masking invalid tokens. Why can it still produce a *wrong* result?

- A. Because constrained decoding cannot guarantee semantic correctness — `{"name": "Alice", "age": -3}` is valid JSON yet wrong.
- B. Because masking invalid tokens corrupts the JSON syntax.
- C. Because constrained decoding only works on prose, not JSON.
- D. Because it raises temperature and randomizes the values.

Answer: A. Constrained decoding guarantees syntactic validity by construction but not semantic correctness — valid syntax can still hold a wrong value. B contradicts its purpose; C and D misstate how it works. **বাংলা:** গঠনগতভাবে বৈধ হলেও মান ভুল হতে পারে — ব্যাকরণ ঠিক করলেই তথ্য সত্য হয় না

Q48. How does constrained decoding actually enforce a schema during generation?

- A. It deletes invalid tokens from the model's vocabulary permanently.
- B. It runs the output through a separate spell-checker after generation.
- C. At each step it sets the logits of grammar-invalid tokens to negative infinity, so after softmax only valid tokens have probability.
- D. It fine-tunes the model on valid JSON before each call.

Answer: C. A grammar/automaton determines valid next tokens; invalid logits go to negative infinity and softmax renormalizes over the valid set. A misdescribes a permanent change; B is post-hoc, not decoding-time; D involves weight updates. **বাংলা:** প্রতিটি ধাপে অবৈধ টোকেনের logit মাইনাস ইনফিনিটি করা হয়, softmax-এ কেবল বৈধ টোকেন টিকে থাকে

Q49. Which statement best distinguishes native tool calling from prompt-based tool calling?

- A. Function schemas are pasted as text into the system prompt and parsed from the model's text output.
- B. Function schemas are provided via a separate structured interface without consuming prompt tokens, and the model emits a special tool-call token.
- C. The model executes the external tool inside its own forward pass.
- D. The application chooses the tool and the model only formats the final answer.

Answer: B. Native tools pass schemas out-of-band (no prompt tokens); the model, trained to select tools, emits a special tool-call token (slide 245 update). A describes *prompt-based* tools; C is impossible (the application executes); D inverts who chooses. **বাংলা:** Native টুলে স্কিমা আলাদা চ্যানেলে যায় (টোকেন লাগে না), মডেল বিশেষ tool-call টোকেন পাঠায়

Q50. In the five-step function-calling procedure, who actually executes the tool, and what does the model do afterward?

- A. The model executes the tool internally and stops.
- B. The user executes the tool manually and the model never sees the result.
- C. The application executes the tool, returns the result to the model, and the model continues generating the final answer.
- D. The tool executes itself and writes directly to the user.

Answer: C. The application runs the tool, feeds the result back, and only then does the model produce the final answer. A is impossible — the model only emits the intention; B and D remove the model from synthesizing the result. **বাংলা:** অ্যাপ টুল চালায়, ফল মডেলকে ফেরত দেয়, তারপর মডেল চূড়ান্ত উত্তর তৈরি করে

Q51. Why does the lecture say native tool schemas consume "no tokens"?

- A. Because the schemas are compressed into a single token.
- B. Because the schemas travel through a separate structured metadata interface rather than the tokenized prompt.
- C. Because the model ignores the schemas entirely.
- D. Because tokens are only counted for the user message.

Answer: B. Native schemas are passed out-of-band via a metadata channel, not the tokenized prompt, so they cost no context-window tokens. A invents compression; C contradicts that the model uses them to select tools; D misstates token counting. **বাংলা:** স্কিমা টোকেনাইজড প্রম্পটে নয়, আলাদা মেটাডেটা চ্যানেলে যায় — তাই কোনো টোকেন খরচ হয় না

Q52. Which statement best distinguishes structured output from function calling?

- A. Structured output executes external tools; function calling only formats text.
- B. They are the same feature under two names.
- C. Structured output requires native tools; function calling forbids JSON.
- D. Structured output enforces a response format; function calling lets the model emit a tool invocation the application executes.

Answer: D. Structured output says "answer in this shape"; function calling says "emit a tool call (which the app runs)." A inverts the two; B denies a real distinction; C fabricates constraints. **বাংলা:** Structured output ফরম্যাট মানায়; function calling টুল-কল পাঠায় যা অ্যাপ চালায়

Topic: Prompt Evaluation and Debugging (Section 3.8)

Q53. The lecture slogan "garbage context, garbage outputs" belongs to which debugging step, and what does it imply?

- A. Identify the failure mode — it implies you should guess the error type.
- B. Check context quality — many prompt failures come from poor context management (buried instructions, irrelevant history, confusing examples).
- C. Add structure — it implies you should always add a JSON schema first.
- D. Test systematically — it implies running the prompt once is enough.

Answer: B. "Garbage context, garbage outputs" is the *check-context-quality* step: buried or missing instructions, irrelevant history, or confusing examples cause failures. A, C, and D map the slogan to the wrong steps. **বাংলা:** "garbage context, garbage outputs" — কনটেক্সটের মান যাচাইয়ের ধাপ; খারাপ কনটেক্সটই অনেক ব্যর্থতার কারণ

Q54. Why does the lecture insist on measuring prompts on a fixed test set rather than judging "by feel"?

- A. Because measuring is slower and therefore more rigorous.
- B. Because without a fixed evaluation set, confirmation bias wins and you cannot tell whether a change truly helped.
- C. Because a test set permanently improves the model's weights.
- D. Because feel-based judgment always underestimates accuracy.

Answer: B. You cannot *guess* whether a prompt is better; a fixed test set prevents confirmation bias and enables regression testing. A misstates the rationale; C involves weight changes (false); D is an unfounded absolute. **বাংলা:** নির্দিষ্ট টেস্ট সেট ছাড়া confirmation bias জেতে — অনুমান নয়, মাপতে হবে

Q55. "If a prompt breaks easily, it is not production-ready." This slogan most directly supports which debugging step?

- A. Improve instructions — by adding a persona.
- B. Identify the failure mode — by labeling the error.
- C. Test systematically — across different phrasings, edge cases, multi-turn conversations, and varying context lengths.
- D. Check context quality — by pruning history.

Answer: C. A prompt that breaks under varied phrasings, edge cases, or context lengths is not production-ready — that is the *test-systematically* step. A, B, and D are other (earlier) steps in the workflow. **বাংলা:** বিভিন্ন phrasing/edge case/দৈর্ঘ্যে টেস্ট করা — সহজে ভেঙে গেলে তা production-ready নয়

Q56. A model-as-judge gives a long, elaborate answer a higher score than a short, correct one. Which bias is this, and what is the consequence for automated prompt optimization?

- A. Position bias; it makes optimization favor whichever answer comes first.
- B. Self-preference bias; optimization would copy the judge's writing style.
- C. Verbosity bias; optimization would then drift toward producing longer answers rather than truly better ones.
- D. Prompt injection; optimization would follow instructions hidden in the content.

Answer: C. Preferring longer answers regardless of quality is verbosity bias; an optimizer using this judge would optimize toward length, not true quality. A, B, and D name real but *different* model-as-judge pitfalls. **বাংলা:** লম্বা উত্তরকে বেশি নম্বর দেওয়া = verbosity bias; অপ্টিমাইজার তখন দৈর্ঘ্যের পেছনে ছুটবে, মানের নয়

Topic: Automated Prompt Optimization (Section 3.9)

Q57. Which statement best captures the spirit of automated prompt optimization?

- A. It turns prompting into a creative art that resists measurement.
- B. It turns prompting into an optimization problem: generate, evaluate, select, repeat.

- C. It replaces the model's weights with optimized prompts.
- D. It removes the need for any evaluation metric.

Answer: B. Automated optimization treats prompting as an optimization loop — generate (mutation/crossover), evaluate, select, repeat until convergence. A is the opposite framing; C confuses prompts with weights; D contradicts its dependence on metrics. **বাংলা:** প্রম্পটিংকে অপ্টিমাইজেশন সমস্যা বানায়: জেনারেট → মূল্যায়ন → বাছাই → পুনরাবৃত্তি

Q58. In the optimization loop, what does the "mutation" step do?

- A. It executes external tools to verify answers.
- B. The model creates many prompt variations by rewriting, adding or removing constraints, rephrasing, or inserting examples.
- C. It permanently changes the model's parameters.
- D. It selects the single best prompt and stops.

Answer: B. Mutation generates prompt variants via the model (rewriting, adding/removing constraints, inserting examples); crossover optionally combines strong parts. A is tool use; C involves weights; D is the *select* step. **বাংলা:** Mutation = মডেল দিয়ে নানা প্রম্পট-ভেরিয়েন্ট তৈরি (পুনর্লিখন, বাধা যোগ/বাদ, উদাহরণ ঢোকানো)

Q59. Why is automated prompt optimization "only as good as the evaluation metric"?

- A. Because a weak metric can be gamed, so the optimizer will optimize toward the metric's flaws instead of true quality.
- B. Because the metric replaces the test set entirely.
- C. Because metrics are irrelevant once mutation begins.
- D. Because a strong metric removes the need for any test set.

Answer: A. The loop optimizes whatever the metric rewards; a weak or biased metric gets gamed, steering toward its flaws (for example a biased model-as-judge). B, C, and D all understate or dismiss the metric's central role. **বাংলা:** দুর্বল মেট্রিক ফাঁকি খায়, তাই অপ্টিমাইজার সত্যিকারের মান নয়, মেট্রিকের ত্রুটির দিকে ছোটে

Q60. The automated-prompt-optimization loop maps cleanly onto evolutionary search. Which correspondence is correct?

- A. Generation = fitness function; evaluation = mutation; selection = crossover.
- B. Generation = mutation/crossover; evaluation = fitness function; selection = survival.
- C. Generation = survival; evaluation = mutation; selection = fitness function.
- D. Generation = selection; evaluation = survival; selection = mutation.

Answer: B. Generation is mutation/crossover, evaluation is the fitness function, and selection is survival of the best — the standard evolutionary template. A, C, and D scramble these roles. **বাংলা:** জেনারেশন = mutation/crossover, মূল্যায়ন = fitness, বাছাই = survival

---

Answer Key

Q#	Ans	Q#	Ans	Q#	Ans	Q#	Ans
Q1	B	Q16	C	Q31	A	Q46	B
Q2	C	Q17	D	Q32	A	Q47	A
Q3	D	Q18	A	Q33	D	Q48	C
Q4	A	Q19	C	Q34	A	Q49	B
Q5	C	Q20	D	Q35	C	Q50	C
Q6	D	Q21	A	Q36	D	Q51	B
Q7	A	Q22	B	Q37	D	Q52	D
Q8	C	Q23	A	Q38	B	Q53	B
Q9	A	Q24	C	Q39	C	Q54	B
Q10	D	Q25	D	Q40	B	Q55	C

Q#	Ans	Q#	Ans	Q#	Ans	Q#	Ans
Q11	C	Q26	A	Q41	C	Q56	C
Q12	A	Q27	C	Q42	D	Q57	B
Q13	D	Q28	B	Q43	B	Q58	B
Q14	A	Q29	D	Q44	D	Q59	A
Q15	B	Q30	A	Q45	D	Q60	B

Answer Distribution

Letter	Count
A	15
B	15
C	15
D	15

বাংলা ব্যাখ্যা: মোট ৬০টি MCQ উপরের উত্তর-কী দিয়ে নিজেকে যাচাই করো; প্রতিটি ব্যাখ্যায় কেন সঠিক ও কেন একটি distractor ভুল — দুটোই দেওয়া আছে