COMBIGRAPH-VIS: A CURATED MULTIMODAL OLYMPIAD BENCHMARK FOR DISCRETE MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Progress on math-reasoning benchmarks such as GSM8K and MATH500 has eroded their ability to discriminate among models with diverse capabilities, motivating harder tests that separate capabilities more sharply. We introduce CombiGraph-Vis, an Olympiad-style benchmark of 1,135 short-answer, multiple-choice, and yes/no problems drawn from the first and second rounds of the Iranian Informatics Olympiad, with 35% multimodal items containing images. The benchmark focuses on discrete mathematics with a computer-science accent, combinatorics, algorithmic techniques, and graph theory, along with probability, discrete and computational geometry, combinatorial game theory, formal languages and automata, conceptual data structures, and logic-driven puzzles. To make the benchmark more functioning, we include corrected official solutions, fixed via an agentic pipeline with human oversight, plus clear, classroom-style rewrites using Gemini 2.5 Pro that elaborate on terse reasoning. Our evaluation suite covers standard accuracy across formats and includes protocols for test-time scaling and self-verification spanning model families from Google, OpenAI. On single-sample accuracy, models range from 16.15% (gemma-3-4b-it) to 78.00% (gpt-5), demonstrating strong separation compared to saturated benchmarks. We release all data, corrected solutions, classroom-style rewrites, evaluation code, and synthetic technique labels under an open-source license to facilitate advances in multimodal algorithmic reasoning. We share all of our code and data publicly in the paper's Github repository: https://github.com/combigraphviz2025/combigraph-viz

1 INTRODUCTION

Mathematical reasoning benchmarks like GSM8K(Cobbe et al., 2021) and MATH(Hendrycks et al., 2021) now show ceiling effects, with leading models achieving 95-96% accuracy. This progress, while substantial, has reduced the discriminative power of these benchmarks for distinguishing capabilities among frontier systems. Existing multimodal mathematical benchmarks like MathVista(Lu et al., 2024) and MathV(Wang et al., 2024) provide broad domain coverage but often lack the depth needed to assess discrete mathematical reasoning skills. Competition-level datasets present complementary limitations: CHAMP(Mao et al., 2024) offers detailed annotations but covers a broad range of mathematical topics without focused depth in discrete domains and only contains 270 samples. OMNI-MATH(Gao et al., 2024) adapts proof-based competition problems for final-answer evaluation, where proof-based problems (originally designed to assess reasoning processes) are evaluated by final answers alone, bypassing their intended assessment focus(Mahdavi et al., 2025).

Discrete mathematical reasoning, spanning combinatorics, logical deduction, graph theory, and algorithmic techniques, remains underrepresented in current multimodal benchmarks. These problems require mathematical insight that goes beyond pattern matching: determining optimal arrangements in combinatorial puzzles, identifying structural properties in graph diagrams, and solving logical constraints across visual representations. To address this gap, we introduce CombiGraph-Vis, a multimodal benchmark of 1,135 discrete mathematics problems designed to evaluate reasoning capabilities across combinatorics, logic, graph theory, and algorithmic techniques and closely related areas.

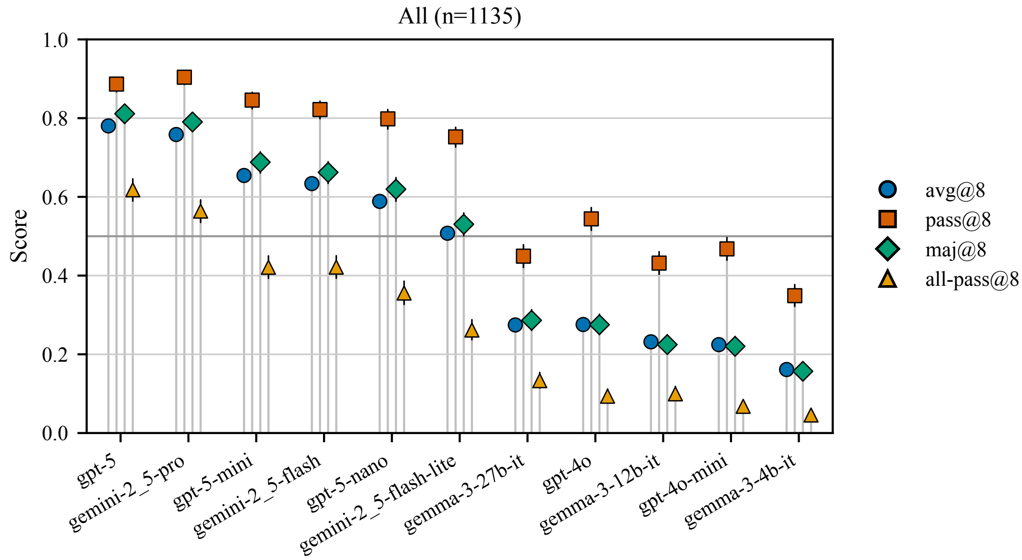

Figure 1: **Per-model evaluation across all 1135 problems in our dataset**. For each model, four horizontal tracks show `avg@8`, `pass@8`, `maj@8`, and `all-pass@8`.

CombiGraph-Vis sources problems from Iranian Informatics Olympiad competitions (both first and second rounds), which concentrate on discrete mathematics across four core domains: combinatorics and counting principles, logical and puzzle reasoning, graph theory, and algorithmic techniques. These problems also include probability, geometry, and game theory components. The problems are concise yet sophisticated, often requiring case analysis, invariant identification, logical deduction, and combinatorial constructions. Importantly, 35% include essential visual components (graphs, grids, geometric figures, logical diagrams) whose structure is integral to the solution, yielding short, verifiable answers across multiple formats(He et al., 2024; Wu et al., 2023; Lu et al., 2024).

To ensure reliability, we systematically correct and enhance the original solutions through automated error detection, cross-validation, and expert review, followed by clear explanatory rewrites. We provide technique categories across key areas of discrete mathematics to enable detailed analysis. All problems are translated from Persian to English with careful attention to preserving both textual and visual content integrity. Evaluation across leading model families reveals substantial performance gaps, with single-sample accuracy ranging from 16.15% (gemma-3-4b-it) to 78.00% (gpt-5) as indicated in Figure 1. Performance varies significantly across problem formats and visual vs. text-only conditions. This work contributes a discrete mathematics benchmark with verified solutions, systematic evaluation revealing model limitations, and complete open-source release.

2 Related Work

Mathematical Reasoning Benchmarks. GSM8K introduced 8,500 grade school math word problems with verification-based training, demonstrating that step-by-step solutions improve both accuracy and reliability(Cobbe et al., 2021). MATH scaled this approach to high school competition mathematics with 12,500 problems across algebra, geometry, number theory, and other domains(Hendrycks et al., 2021). Methodological advances complemented these datasets: chain-of-thought prompting enabled explicit reasoning steps(Wei et al., 2022), while self-consistency enhanced reliability through majority voting over multiple solution paths(Wang et al., 2023). Competition-focused datasets followed with CHAMP providing 270 problems with rich concept-level annotations(Mao et al., 2024) and OMNI-MATH aggregating 4,428 Olympiad-style problems from international competitions across over 33 mathematical sub-domains(Gao et al., 2024).

Visual Mathematical Reasoning. Visual mathematical reasoning benchmarks address problems where images contain essential information for solving mathematical questions. Domain-specific approaches include GeoQA with 5,010 geometric problems requiring diagram interpretation(Chen et al., 2021) and Conic10K with 10,861 conic section problems providing formal symbolic representations(Wu et al., 2023). Comprehensive collections followed: MathVista combines 6,141 visual math problems from 28 existing datasets spanning geometry, statistics, and algebraic reasoning(Lu et al., 2024), MATH-V curates 3,040 competition problems requiring visual context understanding across 16 mathematical disciplines(Wang et al., 2024), and OlympiadBench extends beyond mathematics with 8,476 bilingual multimodal problems covering both mathematics and physics from international competitions(He et al., 2024).

General Multimodal Reasoning. General multimodal reasoning benchmarks evaluate capabilities beyond mathematical domains. MMMU targets expert-level understanding with 11,500 college questions spanning art, business, science, health, humanities, and social science(Yue et al., 2024b), while MMBench provides systematic evaluation across 20 ability dimensions with 3,000+ multiple-choice questions(Li et al., 2024). Knowledge-intensive approaches include A-OKVQA with 25,000 questions requiring both visual understanding and world knowledge(Schwenk et al., 2022) and CLEVR-Math with 10,000 synthetic questions testing systematic combination of arithmetic operations in visual contexts(Liu et al., 2022).

Evaluation Methods and Robustness. Advanced evaluation methods examine solution quality and reasoning stability beyond final answer accuracy. We-Math introduces a diagnostic framework that decomposes 15,000 mathematical problems by knowledge concepts and evaluates models across four categories: insufficient knowledge, inadequate generalization, complete mastery, and rote memorization(Qiao et al., 2025). DynaMath focuses on robustness evaluation by generating multiple variants of each seed problem, creating 501 base problems with over 5,000 variations to test consistency across input perturbations(Zou et al., 2025), while MPBench provides a meta-evaluation framework for visual mathematical reasoning, testing models' abilities in step checking, solution aggregation, and guided step selection across 1,000 competition problems(Pan et al., 2025).

Solution Assessment. Evaluating open-ended mathematical solutions presents unique challenges requiring specialized assessment frameworks. HARP compiles 3,000 short-answer competition problems from prestigious contests, providing multiple human solution strategies and reference answers to enable comprehensive evaluation(Yue et al., 2024a), while U-MATH targets university-level mathematical reasoning with 1,100 problems spanning calculus, linear algebra, and advanced topics, introducing a meta-evaluation framework that assesses the quality of LLM-based grading systems(Chernyshev et al., 2025). CombiGraph-Vis combines these threads: discrete math problems with images, short checkable answers, and detailed solution steps. It emphasizes combinatorics, logic, graph theory, and algorithmic techniques, and pairs verified solutions with evaluation that reports results by format and modality.

3 CombiGraph-Vis Dataset

Discrete mathematical reasoning requires analyzing combinatorial structures, proving graph properties, and constructing algorithmic solutions: capabilities that current models struggle with. CombiGraph-Vis addresses these evaluation needs with 1,135 competition-level problems sourced from Iranian Informatics Olympiad rounds; it covers 13 domains from basic counting principles to advanced topics like combinatorial game theory and computational geometry. The benchmark provides three problem formats: 884 short-answer problems requiring precise mathematical responses, 157 multiple-choice problems testing conceptual understanding, and 94 binary problems demanding logical conclusions (see Table 1 for detailed statistics). Visual components appear in 406 problems (36%), featuring graphs, grids, diagrams, and puzzle boards. Structural interpretation is essential for solving these problems. Each problem includes verified solutions and systematic technique categorization across combinatorics, graph theory, algorithmic reasoning, and logical puzzle solving, enabling detailed analysis of model capabilities in discrete mathematical domains.

3.1 Data Collection

Building a multimodal discrete mathematics benchmark from competition sources requires careful handling of changing formats over time. The Iranian National Olympiad in Informatics changed format significantly between the 5th and 34th competitions, shifting from mainly multiple-choice

Category	Count	% of Total	With Images
All Problems	**1,135**	**100.0**	**406 (35.8%)**
Short-answer	884	77.9	321 (36.3%)
Multiple-choice	157	13.8	49 (31.2%)
Yes/No	94	8.3	36 (38.3%)

Table 1: CombiGraph-Vis dataset statistics.

problems to include short-answer and yes/no formats. We collected problems from first rounds (competitions 534) and selected second rounds (24th, 25th, 26th, 30th, 32nd) that contained our target problem types. Competition PDFs provided the primary source material, with Opedia.ir used for validation and filling gaps.

Adapting Persian materials for international use involved several challenges. Translation alone was insufficient: many problems had interconnected contexts requiring shared definitions or multi-part scenarios. Contextual field annotation solved this by preserving problem dependencies while enabling standalone evaluation (see Figure 3 for an illustration). Visual elements needed quality assessment and recreation when Persian text or poor resolution made them inaccessible. During curation, we discovered that many originally multiple-choice problems actually functioned independently of their provided options. An agentic classification workflow now distinguishes ”standalone” problems from genuinely ”choice-dependent” ones, expanding format options. Figure 2 illustrates this distinction with representative examples from our dataset.

3.2 Data Curation Process Using Agentic Workflows

We applied agentic workflows with human-in-the-loop to fix existing errors in the dataset during the data curation phase. Our initial analysis identified three distinct error categories with different patterns requiring specialized detection approaches:

1. **Conversion errors** from automated PDF parsing, including issues with mathematical notation, formatting artifacts, and character encoding problems;
2. **Translator/annotator errors** ranging from typos to semantic mistranslations that compromised problem clarity;
3. **Original source errors** from OCR processes, which occurred frequently as many archived competition PDFs came from OCR conversion of paper documents rather than original digital files.

3.2.1 First Phase: Problem Validation

We developed a two-phase filtering process using agentic workflows to detect mistakes in problems and solutions. Our first phase uses an agentic workflow that generates validation reports through three specialized critics (Figure 4). Each critic has access to the problem context (if any): problem text, English solution, original Persian problem and solution, answer choices, correct option, and final answer.

The three critics operated as:

1. **Typo/Clarity Critic** compares English translations with original Persian text to identify typos and clarity issues;
2. **Logical Soundness Critic** verifies reasoning consistency and computational accuracy;
3. **Final Answer Match** checks whether the final answer derived in the solution text matches the stored final answer entry.

We run this workflow three times independently for each problem to generate three validation reports. We then use an aggregator stage that applies majority voting to synthesize the three reports into structured JSON output with multiple diagnostic fields. Complete implementation details for the first phase are provided in Algorithm 2 (Appendix A). For filtering purposes, we use the Overall Error Severity score using a 5-point scale which is defined as follows:

Choice-Dependent Problem

A calculating machine has an internal memory called M. This machine can calculate an expression by performing the following instructions:

- `Add X`: Adds the value of X to the value of M and stores the result in M.
- `Mul X`: Multiplies the value of X by the value of M and stores the result in M.

In the above instructions, X can be an integer or a variable. Assume the initial value of M is zero.
Example: The following instructions, from left to right, calculate the expression $ax + 5$: `Add a`, `Mul x`, `Add 5`.
Which of the following expressions cannot be calculated by this machine?

1. $ax^2 + bx + c$
2. $(a + b)xy + ya$
3. $(ax + by)(a + b)$
4. $3x^5 + 1$
5. All these expressions can be calculated

Standalone Problem (Originally Multiple-Choice)

We have written numbers 1 to 78 clockwise on a circle. We select the number 1 as the current number and repeat the following operations until only one number remains on the circle:

- If the current number is x, remove it from the circle, add one unit to the x next numbers clockwise on the circle, and select the number after that (two places clockwise from the removed number) as the current number.

Note that if the number of remaining numbers on the circle is less than 3, one or more numbers might have more than one unit added to them.
What is the remainder when the number that finally remains on the circle is divided by 5?

Original choices:

1. 0
2. 1
3. 2
4. 3
5. 4

(now used as short-answer format)

Figure 2: Examples of choice-dependent vs. standalone problems. The first requires analyzing provided options to determine impossibility, while the second has a unique numerical answer independent of choices.

- **1 (No issues):** Clear and correct overall
- **2 (Minor issues):** Small problems with no impact on meaning
- **3 (Moderate issues):** Multiple clarity problems or one significant issue
- **4 (Major issues):** Significant contradictions or error patterns that likely invalidate the solution
- **5 (Critical failure):** Pervasive issues or fatal flaws making the pair unusable

We checked the generated reports for a handful of cases and detected systematic patterns where problems flagged with ”major issues” typically contained only minor typos, while those marked ”critical

Context-Dependent Problem

Context: Consider the following definition for the next three questions: An $m \times n$ table where each cell contains an integer is called a 'counting table' if the absolute difference of the numbers written in any two adjacent (row-wise or column-wise) cells is exactly one. As an example, the table below is a 2×3 counting table.

2	3	2
3	2	1

Question: A counting $m \times n$ table, with all its cells filled, is given. We want to reveal the numbers in a minimum number of its cells (their numbers become known to us) so that we can deduce the numbers in the remaining cells. In what range does this minimum lie?

1. 1 or 2
2. $[3, m+n-1]$
3. $[\frac{mn}{2}, m+n]$
4. $[\frac{mn}{2}, mn-1]$
5. Exactly mn

Figure 3: Example of a context-dependent problem requiring shared definitions from a multi-part scenario. The contextual field preserves the counting table definition needed to understand the question.

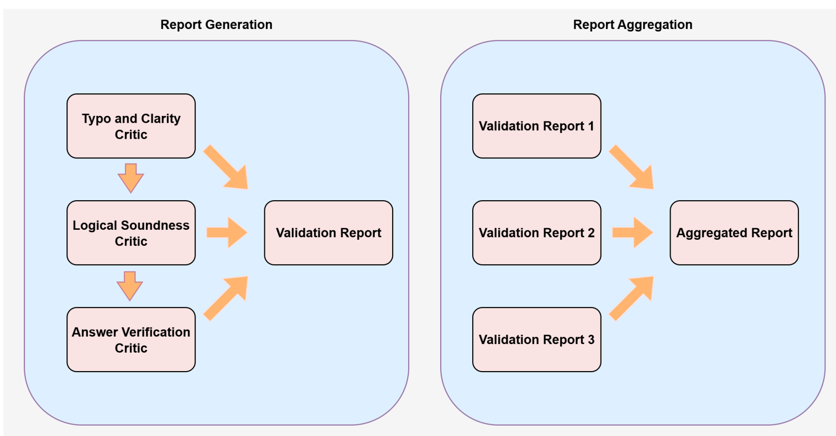

Figure 4: Agentic validation pipeline for quality assurance. The process consists of two main phases: Report Generation with three specialized critics (Typo/Clarity, Logical Soundness, Answer Verification) running in parallel, followed by Report Aggregation that synthesizes multiple validation reports through majority voting to produce final quality assessments.

failure" often had single correctable errors. We filtered all cases with severity scores above 1 for the second validation and error correction phase, accepting this conservative threshold to minimize false negatives while managing the high false positive rate we observed.

3.2.2 SECOND PHASE: AUTOMATED ERROR RESOLUTION

Many problems flagged in the first phase came from common parsing errors and misunderstanding brief solutions by the model, not actual errors from the original sources. We found recurring problems: equation parsing errors (e.g. binomial notation converted to fractions), translation mistakes (choice permutations, typos), and false positives where models struggled with the concise original solutions.

We developed an error resolution workflow that categorizes errors using patterns identified from first-phase validation logs. By analyzing validation reports, we distinguished between errors introduced by our pipeline versus those present in original sources. Algorithm 1 shows the high-level stages of the workflow. The workflow handles three error types with different approaches: pipeline errors (parsing/conversion problems) receive direct fixes for notation and formatting issues; potential source errors trigger a solution expansion phase where we rewrite the original brief solution into detailed, step-by-step explanations under the assumption that the final answer is correct; and image-understanding issues are escalated to human review.

Algorithm 1 Error Resolution Workflow

Require: Problem data d, validation reports from first phase
Ensure: Fixed problem data or human intervention report
Load and aggregate validation findings
Classify error type: pipeline, source, or image-understanding
if pipeline error **then**
    Apply targeted fixes (notation, formatting, choices)
else if source error **then**
    Engage with solution for deeper analysis
    Reclassify with expanded context
else if image-understanding issue **then**
    Escalate to human review
end if
if automated fix required **then**
    repeat
        Plan surgical edits with constraints
        Apply fixes and validate with 5 consecutive successes
    until quality threshold met or budget exceeded
end if
return fixed data or human intervention report

We observed that most original source errors occurred in the solutions rather than in problem statements or final answers. To address this, our solution expansion approach rewrites brief original solutions into detailed, step-by-step explanations while assuming the correctness of the final answers. For automated fixes, we only edit data classified as having ”Minor, Fixable Issues” using predefined criteria in our prompts - where the mathematical approach is sound but contains localized errors like typos, calculation mistakes, or unclear presentation. We avoid editing cases with ”Major Logical Flaws” where the core method is fundamentally incorrect. The workflow can edit all data fields (problems, solutions, answers) while preserving image file names and paths. The workflow validates fixes through an automated iterative process: a validator stage checks each proposed fix against the original detected issues using the problem information (stem, solution, context, final answer) and outputs from previous stages, and the system requires the same fix to pass validation 5 consecutive times before accepting changes. This is because each stage is an LLM call and it has non-deterministic behavior and repeated calls can lead to different outputs, hence, repeating the same validation stage in a loop makes it more reliable. If any validation fails, the success counter resets and the system generates a new fix plan. After the workflow completes, cases that do not require human intervention are reviewed by a human who accepts or rejects the automatic fix and manually corrects any remaining issues. Cases flagged as requiring human intervention are manually fixed by the human reviewer. Complete implementation details are provided in Algorithms 3 and 4 (Appendix A).

3.2.3 Technique Labels and Taxonomy

To enable fine-grained analysis of mathematical reasoning capabilities, we applied technique labeling based on the official Iranian Informatics Olympiad curriculum. Each problem receives hierarchical labels following a three-level taxonomy: Topic → Sub-topic → Sub-sub-topic (e.g., Combinatorics → Counting Foundations → Stars & bars). We use a single prompt that assigns labels based on techniques that explicitly appear in solution steps. The taxonomy covers 13 major topics spanning discrete mathematics with 89 distinct sub-sub-topic labels that capture precise mathematical approaches used in solutions. This fine-grained labeling enables researchers to analyze model performance across specific techniques, identify capability gaps, and design targeted evaluation protocols. The complete hierarchical taxonomy and labeling prompt are provided in Appendix C.

4 Task Formats and Verification Protocol

We evaluate models by generating eight solutions per problem using a chain-of-thought prompt that instructs models to produce step-by-step reasoning and wrap the final answer in `\boxed{}` format (Appendix C.2). For choice-dependent multiple-choice problems, we include the answer choices in the prompt to ensure the model selects from the provided options. To parse the the final answer from the model's output, we use a simple regex pattern that matches the `\boxed{}` format. If all of the choices for that specific problem were numerical/algebraic expressions, we used the Math-VerifyKydlek & Gandenberger (2024) library to check if the extracted answer is equivalent to the final answer. In case the generated solution didn't follow the instruction and didn't wrap the final answer in `\boxed{}`, or the choices were not numerical/algebraic expressions, we offloaded the task to an LLM (Gemini 2.5 Flash) to extract the final answer. In the prompt, we asked the model to extract the final answer's raw value, and the matching choice (if any) and the standardized form of the final answer (in case the choices were not numerical/algebraic expressions and the final answer matched one of the choices). We then checked if the extracted answer is equal to the final answer or the extracted choice is equal to the correct option.

5 Results

Across all evaluation settings, we observe clear separations between model families, with top-tier models achieving strong but far from saturated accuracy, mid-tier models trailing substantially, and lightweight/open-weight models far behind. Accuracy drops on image-tagged items compared to text-only items, revealing persistent gaps in visual mathematical understanding. Multiple-choice behavior shows a pronounced discrepancy between standalone and among-choices accuracy, indicating that models are often lured by wrong answers deliberately crafted in competition settings.

Overall Performance Top-level results are summarized in Table 2 (cf. Figure 1). Top-tier models reach single-sample averages around 75–78% while mid-tier and lightweight/open-weight models lag by 20–40+ points depending on the evaluation setting. This broad dispersion persists across formats and modalities, confirming that CombiGraph-Vis is not saturated: even the strongest models leave substantial headroom while weaker models remain far from ceiling. The per-model tracks (avg@8, pass@8, maj@8, all-pass@8) further reinforce clear separations among model families.

Table 2: avg@8 reported across evaluations settings. Best performance in each slice is highlighted.

		Images		Multiple-Choice			
Model	All	Yes	None	Standalone	Choice-Dep.	Yes/No	Second Round
gemini-2_5-flash	63.4	50.9	70.3	63.4	56.9	74.1	50.4
gemini-2_5-flash-lite	50.8	33.8	60.2	49.1	50.6	66.4	30.2
gemini-2_5-pro	75.8	66.9	**80.8**	75.7	72.9	**81.9**	71.6
gemma-3-12b-it	23.2	17.5	26.3	21.2	31.1	28.3	13.7
gemma-3-27b-it	27.5	20.1	31.6	25.0	38.5	32.4	12.6
gemma-3-4b-it	16.1	12.1	18.4	13.6	15.9	40.6	9.7
gpt-4o	27.6	20.4	31.6	24.5	31.4	49.9	15.9
gpt-4o-mini	22.5	16.9	25.5	18.9	25.2	50.8	14.6
gpt-5	**78.0**	**68.2**	**83.5**	**77.7**	**81.2**	75.7	**75.6**
gpt-5-mini	65.4	53.9	71.8	67.8	69.0	37.4	59.9
gpt-5-nano	58.9	43.5	67.5	61.1	55.4	44.4	46.3

Modality Gap Table 2 shows consistent drops on image-tagged items relative to text-only problems. For top-tier models, the gap from no-image to image conditions is typically 14–16 percentage points (e.g., 83.5% → 68.2% and 80.8% → 66.9%), and for mid-tier models it can approach 20 points. This indicates that parsing and reasoning over structured visualsgraphs, grids, geometric diagramsremain central bottlenecks, materially impacting overall accuracy.

Standalone vs Among-Choices on MC (short-answer setting) As discuseed, we convert MC problems to short-answer by removing options. For each problem and model we compute: (i) **Standalone avg@8** = mean correctness over 8 samples; and (ii) **Among-Choices avg@8** = mean fraction of samples whose final answer lies among the original (now-hidden) options (not necessarily correct).

Table 3: Standalone vs Among-Choices (avg@8). Δ = (Among-Choices − Standalone) in percentage points.

Model	Standalone (%)	Among-Choices (%)	Δ (pp)
gpt-5	77.7	92.0	14.3
gemini-2_5-pro	75.8	90.0	14.3
gpt-5-mini	67.8	85.4	17.6
gemini-2_5-flash	63.5	83.7	20.3
gpt-5-nano	61.1	82.9	21.8
gemini-2_5-flash-lite	49.2	73.1	23.9
gemma-3-27b-it	25.0	70.4	45.5
gpt-4o	24.6	64.1	39.6
gemma-3-12b-it	21.3	65.4	44.2
gpt-4o-mini	19.0	60.4	41.5
gemma-3-4b-it	13.6	57.5	43.9

These large Δ values indicate that models consistently produce answers that coincide with some provided choice but not necessarily the correct one. In competition settings, answer options are deliberately constructed to include plausible distractors; the systematic gap between Among-Choices and Standalone accuracy thus reveals a susceptibility to these traps. In other words, option exposure often steers models toward distractor recognition rather than robust derivation, whereas the standalone format demands genuine solution construction. Moreover, the large Δ values provide strong support for the adoption of our evaluation suite as an RL environment, since the model can potentially learn to avoid the deliberately crafted distractors, an ability that is prerequisite for performing well in competition-level reasoning.

Topic-Level Performance Per-topic accuracies highlight both broad strengths and persistent weaknesses. Top-tier models are strong in combinatorics, number reasoning, and invariants/monovariants, and they show competitive results in computational geometry; probability is especially high for some models (see Table 4). In contrast, graph-theoretic subdomains (e.g., connectivity, matchings) and formal languages expose larger spreads across models, with mid-tier and lightweight/open-weight models struggling markedly. The dispersion suggests that discrete, structure-sensitive reasoning is not uniformly mastered across mathematical domains.

Table 4: Per-model accuracy by topic (%). Best score per topic is highlighted.

Model	Combinatorics	Logical & Puzzle	Algorithms & DS	Graph	Number	Comb. Game	Probability	Comp. Geometry	Invariants	Formal Lang
gemini-2_5-flash	70.1	56.8	55.0	53.3	76.9	55.2	89.8	56.8	63.8	37.5
gemini-2_5-flash-lite	57.9	44.4	43.1	36.8	68.2	36.6	82.8	39.8	57.5	28.1
gemini-2_5-pro	**82.1**	69.4	67.5	70.2	85.8	**69.5**	**91.4**	73.9	87.5	65.6
gemma-3-12b-it	26.5	18.0	16.7	17.8	32.6	27.7	54.7	14.8	18.8	12.5
gemma-3-27b-it	30.7	23.6	22.7	19.3	36.0	25.0	62.5	11.4	17.5	25.0
gemma-3-4b-it	15.3	15.6	14.5	10.9	23.1	20.7	16.4	19.3	10.0	12.5
gpt-4o	29.3	23.4	24.5	25.2	32.1	25.3	55.5	27.3	13.8	15.6
gpt-4o-mini	23.8	18.6	18.1	17.5	30.1	23.2	51.6	23.9	13.8	15.6
gpt-5	81.6	**73.4**	**73.1**	**76.3**	**86.6**	61.6	77.3	**79.5**	**95.0**	**100.0**
gpt-5-mini	70.5	57.1	59.7	64.3	74.4	48.8	77.3	50.0	86.3	81.3
gpt-5-nano	65.5	51.9	49.6	52.4	73.2	39.9	78.3	42.5	81.3	78.1

6 CONCLUSION

Together, our findings indicate that CombiGraph-Vis yields strong separations across model families, exposes enduring multimodal reasoning deficits, and stresses the difference between distractor-sensitive recognition and derivation-based solution. We leverage these observations in the Discussion to analyze error modes and to outline methodological directions for building models that can reliably solve complex, multimodal discrete mathematics problems.

7 LLM USAGE DESCRIPTION

We used LLMs such as gpt-5 and Gemini 2.5 Pro to polish writing, fix grammatical errors and latex alignment issues.

REFERENCES

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric P. Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. *arXiv preprint arXiv:2105.14517*, 2021. URL https://arxiv.org/abs/2105.14517.

Konstantin Chernyshev, Vitaliy Polshkov, Vlad Stepanov, Alex Myasnikov, Ekaterina Artemova, Alexei Miasnikov, and Sergei Tilga. U-math: A university-level benchmark for evaluating mathematical skills in large language models. In Ofir Arviv, Miruna Clinciu, Kaustubh Dhole, Rotem Dror, Sebastian Gehrmann, Eliya Habba, Itay Itzhak, Simon Mille, Yotam Perlitz, Enrico Santus, João Sedoc, Michal Shmueli Scheuer, Gabriel Stanovsky, and Oyvind Tafjord (eds.), *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, pp. 974–1001, Vienna, Austria and virtual meeting, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-261-9. URL https://aclanthology.org/2025.gem-1.77/.

Karl Cobbe, Vlad Lyzhov, Mohammad Bavarian, Michael Kossakowski, Heewoo Chen, Alethea Power, Lukasz Kaiser, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations (ICLR) OpenReview*, 2024. URL https://openreview.net/forum?id=yaqPf0KAlN.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks et al. Measuring mathematical problem solving with the math dataset. In *International Conference on Learning Representations (ICLR) OpenReview*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Hynek Kydlek and Greg Gandenberger. Math-verify: A python library for mathematical expression verification, 2024. URL https://github.com/huggingface/Math-Verify. Version 0.8.0.

Jie Li et al. Mmbench: Is your multi-modal model an all-around player? In *Computer Vision – ECCV 2024*, 2024.

... Liu et al. Clevr-math: A dataset for compositional language, visual and mathematical reasoning. *arXiv preprint arXiv:2208.05358*, 2022. URL https://arxiv.org/abs/2208.05358.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KUNzEQMWU7. ICLR 2024 (oral).

Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant G. Honavar. Brains vs. bytes: Evaluating llm proficiency in olympiad mathematics. In *arXiv preprint arXiv:2501.xxxxx*, 2025. URL https://openreview.net/forum?id=V4RIJxt02s.

Yujun Mao, Yoon Kim, and Yilun Zhou. Champ: A competition-level dataset for fine-grained analyses of LLMs' mathematical reasoning capabilities. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13256–13274, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.785. URL https://aclanthology.org/2024.findings-acl.785/.

xu Zhao Pan, Pengfei Zhou, Jiaxin Ai, Wangbo Zhao, Kai Wang, Xiaojiang Peng, Wenqi Shao, Hongxun Yao, and Kaipeng Zhang. Mpbench: A comprehensive multimodal reasoning benchmark for process errors identification. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 21586–21606, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1112. URL https://aclanthology.org/2025.findings-acl.1112/.

Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma GongQue, Shanglin Lei, YiFan Zhang, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Xiao Zong, Yida Xu, Peiqing Yang, Zhimin Bao, Muxi Diao, Chen Li, and Honggang Zhang. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In *Computer Vision – ECCV 2022*, pp. 146–162. Springer, 2022.

Ke Wang et al. Measuring multimodal mathematical reasoning with math-vision dataset. In *NeurIPS 2024 Datasets and Benchmarks Track*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/1b8a53a4d483589a0b07fdd2a9e4d4b2-Abstract-Datasets_and_Benchmarks.html.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Haoyi Wu, Wenyang Hui, Yezeng Chen, Weiqi Wu, Kewei Tu, and Yi Zhou. Conic10k: A challenging math problem understanding and reasoning dataset. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 6444–6458, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.427. URL https://aclanthology.org/2023.findings-emnlp.427/.

Albert S Yue, Lovish Madaan, Ted Moskovitz, DJ Strouse, and Aaditya K Singh. Harp: A challenging human-annotated math reasoning benchmark. *arXiv preprint arXiv:2412.08819*, 2024a. URL https://arxiv.org/abs/2412.08819.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9556–9567, June 2024b.

Chen Zou, Yixuan Song, Zhen Hu, Yitong Liao, Chunyuan Li, Xun Yang, and Yizhou Wang. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=VOAMTA8jKu.

A Implementation Details

Algorithm 2 Problem Validation Workflow (First Phase)

Require: Problem datum d = (problem, choices, english_solution, context, correct_option, answer_value, crawled_persian_markdown, svg_sources)
Ensure: problem_validation_data

```
reports ← [ ]
for i ← 1 to 3 do
    typo_report ← TypoClarityCritic(d)
    logic_report ← LogicalSoundnessCritic(d)
    answer_report ← AnswerVerificationCritic(d)
    combined_report ← ReportCollector(typo_report, logic_report, answer_report)
    Append(reports, combined_report)
end for
joined_reports ← JoinReportChunks(reports)
validation_result ← FinalAggregator(joined_reports)
return validation_result
```

Algorithm 3 Error Detection and Classification

Require: Problem datum d = (problem, choices, english_solution, context, correct_option, answer_value, crawled_persian_markdown, svg_sources)
Ensure: Classification result agg with fix requirements

```
findings_md ← BuildFindingsText(LoadValidationData(d.id))
reports ← [ ]
for i ← 1 to 3 do
    r ← IssueDetector(d, findings_md)
    Append(reports, r)
end for
reports_md ← JoinIssueReportChunks(reports)
agg ← IssueAggregator(reports_md, d)
if agg.is_original_source_error then
    engagement_md ← SolutionEngager(d, agg.aggregated_report_md)
    src_cls ← IssueDetectorWithEngagement(d, engagement_md)
    src_cls_md ← FormatToMarkdown(src_cls)
    agg ← EngagementReportSynthesizer(agg.aggregated_report_md, engagement_md, src_cls_md)
    if agg.requires_human_intervention then
        return ComposeHumanInterventionReport(agg)
    end if
else if agg.is_image_understanding_issue then
    return ComposeHumanInterventionReport(agg)
end if
return agg                                ▷ Classification result for automated fixing
```

Algorithm 4 Automated Error Resolution and Fixing

Require: Problem datum d, classification result agg from Algorithm 3
Ensure: Fixed problem data or human intervention report

```
fix_plan_md ← FixPlanner(agg.aggregated_report_md, d)
fixed ← Fixer(fix_plan_md, d)
ctx ← UpdateContextWithFixes(fixed)
fixed_md ← FormatFixedData(ctx.fixed_problem_data)
successes ← 0
for t ← 1 to 20 do
    result ← Validator(agg.aggregated_report_md, fix_plan_md, d, fixed_md)
    if result.is_fixed then
        successes ← successes +1
        if successes ≥ 5 then
            break
        end if
    else
        successes ← 0
        fix_plan_md ← RePlanner(agg.aggregated_report_md, result.reasoning, fix_plan_md, d)
        fixed ← Fixer(fix_plan_md, d)
        ctx ← UpdateContextWithFixes(fixed)
        fixed_md ← FormatFixedData(ctx.fixed_problem_data)
    end if
end for
return ComposeAutoFixOutput(d, agg, fix_plan_md, fixed_md)
```

B Prompt Specifications

B.1 Problem Validation Prompts

B.1.1 TypoClarityCritic

TypoClarityCritic Prompt

```
You are a meticulous editor and proofreader, specializing in
    technical and mathematical content. Your sole task is to review a
    given math problem and its solution for **critical surface-level
    errors that fatally impact its meaning or solvability.** If
    available, you will ALSO be provided with inline SVG XMLs as text
    under the placeholder {svg_sources}; you may use their textual
    content (e.g., embedded <text> labels) as additional context.

**Focus ONLY on the following types of fatal errors:**
- **Semantically Significant Typos:** Look for spelling mistakes,
    incorrect variable names (e.g., ’x’ used in one place, ’X’ in
    another), sign/symbol errors (e.g., ’=’ vs ’’, ’<’ vs ’’),
    misplaced decimals, or unit/notation inconsistencies **that change
    the mathematical meaning**. A typo in a variable/symbol is
    critical; a typo in a descriptive word is not, unless it creates
    ambiguity that affects meaning.
- **Explicit Grammar Errors (Meaning-Changing):** Unambiguous
    grammatical mistakes that alter conditions or conclusions (e.g.,
    missing "not", wrong quantifier, singular/plural mismatch that
    changes scope, misplaced "only"). Do not flag
    awkward-but-understandable text.
- **Meaning-Altering Translation Errors:** Mistranslations that
    invert or distort meaning (e.g., "at least" vs "at most", omission
    of "distinct", "positive" vs "non-negative").

**Crucially, you must IGNORE the following:**
```

````
- Minor grammatical errors that do not change the meaning.
- Awkward but understandable phrasing or style.
- Missing or introduced labels/notation for clarity (e.g., A/B
    labels, introducing variables) unless they create a direct
    contradiction.
- References that belong to problem-solution matching (e.g., claims
    of different problem, domain or method differences)  these are out
    of scope for this stage.
- Mathematical rigor, depth of explanation, or solution correctness.

We are not looking for a perfectly written text. We are looking for a
    **functionally correct** text. Only flag an issue if it prevents a
    reasonably skilled person from understanding and solving the
    problem correctly.

**DO NOT:**
- Solve the problem.
- Verify the mathematical logic.
- Check if the final answer is correct.

You will be provided with the problem, its potential choices, the
    provided solution, and possibly a Persian version of the solution
    for reference.

**Problem Data:**
- **Problem:**
  ```
  {problem}
  ```
- **Choices:**
  ```
  {choices}
  ```
- **Provided English Solution:**
  ```
  {english_solution}
  ```
- **Provided Persian Solution (for reference, may be empty):**
  ```
  {persian_solution}
  ```
- **Context (if any):**
  ```
  {context}
  ```

**Optional SVG XMLs (if provided):**
```
{svg_sources}
```

**Important Note on "Context":** The `Context` field, when present,
    contains a shared introduction or definitions for a set of related
    problems. It is a critical part of the problem statement. You must
    also review the context for any typos, grammatical errors, or
    translation issues.

**CRITICAL: Text-Only Analysis:** Base your analysis EXCLUSIVELY on
    the text content. DO NOT use image analysis to detect
    typos/translation errors. Focus only on the written problem
    statement, solution text, and the content inside the provided SVG
    XMLs (if any).
````

```
**Decision rules (apply all):**
- Evidence requirement: For every flagged issue, quote the exact text
    snippet(s) that demonstrate the error.
- Meaning-change threshold: Only flag if the typo/grammar/translation
    issue plausibly changes the mathematical meaning or solvability.
- Notation consistency: Inconsistent variable names/symbols (e.g.,
    'a' vs '', 'x' vs 'X') are errors only if they create ambiguity or
    contradiction in meaning.
- Scope fence: Do not report missing labels, domain mismatches,
    method selection, or any problem-solution matching concerns; these
    belong to a different stage.
- Ambiguity rule: When uncertain, do not flag as fatal. Note the
    ambiguity and rate severity  2.

Review the texts and produce a report in markdown format.

**Output format** (respond ONLY with Markdown; no JSON, no code
    fences, no extra commentary). Use exactly these sections:

# Summary
- 1 2  sentences describing whether there are meaning-changing surface
    errors (typo/grammar/translation).

# Findings
- Comprehensive bullet list of ALL meaning-changing
    typo/grammar/translation errors you identified (do not omit any).
    For each finding, include:
  - The minimal quoted snippet(s) that show the error
  - A one-line justification of how the error changes
      meaning/solvability (alignment with this stages goal)

# Categories
- Bullet list of applicable categories: typo, grammar_error,
    translation_error, other

# Severity
- Rate the overall severity of issues on a scale from 1 (no issues)
    to 5 (worst case). Use this scale:
  - 1: No issues  text is clear and correct at the surface level
  - 2: Minor issues  small/ambiguous issues; no impact on meaning or
      correctness
  - 3: Moderate issues  multiple issues causing intermittent
      ambiguity; meaning mostly intact
  - 4: Major issues  severe ambiguity/errors that likely change
      meaning or solvability
  - 5: Critical failure  pervasive meaning-changing errors make the
      problem/solution unusable
```

B.1.2 LogicalSoundnessCritic

LogicalSoundnessCritic Prompt

```
You are a data integrity specialist. Your task is to check two simple
    things about the problem-solution pair. Your stage goal is ONLY to
    determine whether the solution is seemingly trying to solve the
    same stated problem, and whether the solution explicitly mentions
    that the original problem was changed. You must NOT assess
    solution correctness, judge the method, or evaluate completeness.

**Your Goal:**
```

```
1. **Same Problem Check**: Does the solution appear to be attempting
    to solve the same problem stated, or does it seem to solve a
    completely different problem?
2. **Problem Substitution Check**: Does the solution explicitly
    mention that the original problem was wrong/changed during the
    exam?

**For Goal 1 - Heuristics to detect different problems:**
- Solution discusses completely different mathematical domain (e.g.,
    problem about geometry, solution about number theory)
- Solution addresses fundamentally different question type (e.g.,
    problem asks for proof, solution provides numerical calculation
    for unrelated quantity)
- Solution starts with completely different input parameters with no
    connection to stated problem
- Solutions final answer targets a different object/type than what
    the problem asks for
- Solution relies on constraints or assumptions not present in, or
    contradicting, the problem/context text

**What to IGNORE for Goal 1:**
- Solution is incomplete, brief, or poorly explained
- Solution uses different approach or method than expected
- Solution shows intermediate calculations or introduces helpful
    notation
- Solution quality, mathematical rigor, or level of detail

**For Goal 2 - Look for explicit statements like:**
- "The original problem was incorrect/changed"
- "This problem was modified from the exam version"
- "The exam had an error, so this version solves the corrected
    problem"

**What to IGNORE for Goal 2:**
- Hints or implications without explicit mention of change/error
- General comments about difficulty, ambiguity, or author preference
- Any inference based on images

**Text sources you may use:**
- The written problem statement and solution text
- The `Context` field (if present)
- The inline SVG XMLs (if provided) available under the placeholder
    `{svg_sources}` treat them strictly as text (e.g., read <text>
    labels), not as images

**CRITICAL: Text-Only Analysis:** Base your analysis EXCLUSIVELY on
    textual sources above. DO NOT use image analysis.

You will find the complete problem data in the preceding messages of
    this conversation, including any typo/clarity analysis.

**Decision rules (apply all):**
- Burden of proof: Declare "different problem" only if at least two
    independent, text-based indicators are present. If evidence is
    single, weak, or ambiguous, classify as "same problem" and note
    uncertainties.
- Evidence requirement: Support each indicator with direct text
    quotes/snippets from the problem/solution (and, if helpful, from
    `{svg_sources}`).
- Derived numbers are allowed: Numbers not in the problem but
    plausibly derived from stated inputs are normal and must not be
    used as evidence of mismatch.
```

```
- Notation neutrality: Symbols/labels introduced by the solution (A,
    B, x1, x2) are not evidence of mismatch unless they contradict
    named entities or constraints explicitly defined in text.
- Answer-target check: If the problem asks for X but the solutions
    final target is Y (different type/object), count as one indicator.
- Constraint alignment: If the solution assumes constraints that
    contradict explicitly stated problem/context constraints, count as
    one indicator.
- Ambiguity rule: When uncertain, default to "same problem" (severity
     2) and list the uncertainties explicitly.

Produce a report in markdown format.

**Output format** (respond ONLY with Markdown; no JSON, no code
    fences, no extra commentary). Use exactly these sections and
    structure:

# Summary
- 1 2  sentences stating whether the solution matches the problem and
    whether substitution is explicitly mentioned.

# Findings
- If none, write: None
- Otherwise, for each finding, use this exact template (leave one
    blank line between findings):
  - Finding ID: F1
  - Goal: same_problem_check | substitution_check
  - Indicators: [indicator_1, indicator_2, ...]
    - Choose from: domain_mismatch, question_type_mismatch,
        input_param_mismatch, answer_target_mismatch,
        constraint_contradiction, explicit_substitution_statement
  - Evidence:
    - Problem: "exact quoted snippet from problem"
    - Solution: "exact quoted snippet from solution"
  - Alignment: One sentence explaining how this finding supports the
      stage goal (same_problem_check or substitution_check)
  - Category: mismatch | other

# Categories
- List only those that apply: mismatch, other

# Severity
- One integer 15 using this scale:
  - 1: Matches; no credible indicators
  - 2: Mostly matches; minor/ambiguous inconsistencies
  - 3: Partial match; one credible indicator
  - 4: Likely different problem; two credible indicators
  - 5: Clearly different problem; multiple strong indicators or
      explicit substitution statement
```

B.1.3 AnswerVerificationCritic

AnswerVerificationCritic Prompt

```
You are a data verification agent. Your job is to perform a simple
    but crucial cross-check of the provided data for a math problem.

**Your Goal:**
- Compare the final answer derived in the **Provided English
    Solution** with the official answer recorded in the database
    fields (‘correct_option‘ and ‘answer_value‘).
```

```
- Identify any discrepancies.

**Example Scenarios to Catch:**
- The solution text concludes that "the answer is 12," but the
    `answer_value` is 15.
- The solution text says "Option 3 is correct," but the
    `correct_option` is 2.
- The problem is a yes/no question, and the solution proves "yes,"
    but the `answer_value` is "no."

You will find the complete problem data (problem statement, choices,
    solution, context, images etc.) in the preceding messages of this
    conversation. Your task is to analyze that information. Use the
    images (if any) associated with the problem and solution. Use them
    to understand the context of any text that refers to them.

**Note on "Context":** The `Context` field may contain definitions
    that clarify the nature of the expected answer (e.g., whether it
    should be an integer, a set, etc.). Keep this in mind during your
    verification.


Analyze the `Provided English Solution` to determine the answer it
    produces, and compare it against the `Correct Option Field` and
    `Answer Value Field`. Produce a report in markdown format, stating
    clearly whether there is a mismatch or if the data is consistent.

Output format (respond ONLY with Markdown; no JSON, no code fences,
    no extra commentary). Use exactly these sections:

# Summary
- 1 2  sentences stating "Consistent" or describing the mismatch and
    where it occurred.

# Findings
- Comprehensive bullet list that explicitly identifies the answer
    extracted from the solution text, the databases
    `correct_option`/`answer_value`, and any mismatch. Include minimal
    quotes where helpful.

# Categories
- Bullet list of applicable categories: mismatch, other

# Severity
- Rate the overall severity of verification issues on a scale from 1
    (no issues) to 5 (worst case). Use this scale:
  - 1: No issues  solution and database are consistent
  - 2: Minor issues  small ambiguity; likely consistent
  - 3: Moderate issues  some ambiguity or partial mismatch
  - 4: Major issues  clear mismatch affecting correctness
  - 5: Critical failure  fundamental inconsistency; recorded answer
      and solution contradict
```

B.1.4 FINALAGGREGATOR

FinalAggregator Prompt

```
You are a senior analyst and judge. Your task is to synthesize
    multiple critique reports into a final, structured JSON conclusion
    that details every unique, validated finding.
```

```
**Input:**
You will receive a single markdown string containing the
    concatenated, synthesized reports from each review iteration.

```
{aggregated_report_md}
```

Your Goal:
1. **Synthesize Unique Findings:** Read all reports and identify
    every distinct issue mentioned. Cluster semantically equivalent
    issues across reports into a single candidate finding.
2. **Majority Vote Inclusion:** For each candidate finding, count how
    many distinct critic reports support it. Include a finding in the
    final output only if it is supported by a majority of critic
    reports (ceil(N/2) where N is the number of critic reports
    considered). Discard singletons.
3. **Extract Details for Each Finding:** For each included finding,
    determine its specific `location` (e.g., "Solution, paragraph 3"),
    its `category`, and a specific `severity` score (1-5) for that
    issue alone.
4. **Determine Overall Severity:** Judge the final `overall_severity`
    based on the number, nature, and severity of all included
    findings. A single critical issue might warrant a 5, but a pattern
    of many moderate issues could also indicate a deeply flawed
    problem. Use the following scale for your final judgment:
   - 1: No issues The problem/solution pair appears clear and
       correct overall.
   - 2: Minor issues One or two small problems with no impact on
       meaning or correctness.
   - 3: Moderate issues Multiple problems hindering clarity, or one
       significant issue.
   - 4: Major issues Several significant contradictions or a pattern
       of errors that likely invalidates the solution.
   - 5: Critical failure Pervasive issues, or a single fatal flaw,
       make the pair unusable.
5. **Write Summary Comment:** Provide a high-level, 2-3 sentence
    `summary_comment` of the findings.
6. **Set Final Flag:** Set `is_issue_detected` to `true` if your list
    of findings is not empty.

Adjudication Rubric:
- Validate each critic claim against text: For every claim, cite
    exact text snippets (problem/solution). Ignore image-based claims.
- Label each claim: Validated, Refuted, or Inconclusive. Include a
    brief reason.
- Conflict resolution: When critics disagree, prefer claims with
    stronger, directly quoted textual evidence. Discard claims lacking
    such evidence or relying on images.
- Majority vote rule: Cluster similar claims across critic reports.
    For each clustered issue, compute support_count = number of
    distinct critic reports that raise it. Include only if
    support_count ceil(N/2). Exclude singletons.
- Output policy: Only include majority-supported, Validated findings
    in `aggregated_findings`. Briefly summarize Refuted/Inconclusive
    or non-majority claims in `summary_comment` as adjudication notes.
- Overall severity: Judge holistically from the included findings
    (count, breadth, severity); do not use max-only.
- Ambiguity bias: If no claim can be validated with direct text
    evidence, set `is_issue_detected` to false and `overall_severity`
    to 1, and explain uncertainty in `summary_comment`.

Output Instructions:
```

```
Produce a single, valid JSON object that conforms strictly to the
    schema below. Do NOT add any extra text, markdown formatting, or
    explanations outside of the JSON object.

**JSON Schema for Output:**
```json
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "ProblemValidationOutput",
  "type": "object",
  "properties": {
    "overall_severity": {
      "type": "integer",
      "minimum": 1,
      "maximum": 5,
      "description": "A final judgment on the overall severity,
          considering all findings. Scale: 1=None, 2=Minor, 3=Moderate,
          4=Major, 5=Critical."
    },
    "summary_comment": {
      "type": "string",
      "description": "A high-level, 2-3 sentence summary of the overall
          findings."
    },
    "aggregated_findings": {
      "type": "array",
      "description": "A list of unique, validated issues found in the
          problem/solution pair.",
      "items": {
        "type": "object",
        "properties": {
          "description": {
            "type": "string",
            "description": "A detailed description of the unique issue,
                synthesized from all critic reports."
          },
          "location": {
            "type": "string",
            "description": "The specific location of the issue (e.g.,
                'Problem Statement, paragraph 2', 'Solution, equation
                3')."
          },
          "category": {
            "type": "string",
            "description": "The category of the issue (one of
                'mismatch', 'typo', 'clarity')."
          },
          "severity": {
            "type": "integer",
            "minimum": 1,
            "maximum": 5,
            "description": "The severity of this specific issue, from 1
                (minor) to 5 (critical)."
          }
        },
        "required": ["description", "location", "category", "severity"]
      }
    },
    "is_issue_detected": {
      "type": "boolean",
      "description": "True if any substantive issue is validated,
          otherwise false."
    }
```
```

````
  },
  "required": [
    "overall_severity",
    "summary_comment",
    "aggregated_findings",
    "is_issue_detected"
  ]
}
```
````


B.2 Error Resolution Prompts

B.2.1 IssueDetector

IssueDetector Prompt

```
**Role:** You are an expert forensic analyst for a multi-stage data
    processing pipeline. Your task is to analyze the provided data,
    identify the root cause of discrepancies based on the known
    pipeline, and classify the error.

### How to Determine the True Final Answer

Before classifying an error, you must determine the ground truth for
    the final answer by following this strict hierarchy. This is the
    most critical part of your analysis.

1. **Find the Stated Answer Key:** First, check the
    `crawled_persian_markdown` for an explicit statement of the
    correct option, like "Option X is correct" (` X ł ł`).
2. **The Stated Answer is the Target:**
   *  If an explicit option is stated, find its corresponding
       **value** from the Persian `choices` list. This value is the
       **intended correct answer (the ground truth)**.
   *  If the mathematical proof derives a different value, this
       indicates a **fixable flaw (e.g., a typo, calculation error, or
       encoding issue) within the proof**. Your task is to assume the
       stated answer is correct and identify the flaw in the proof.
3. **Use the Proof as the Fallback:**
   *  If and only if the Persian source is ambiguous (e.g., "Option ?
       is correct"), you must then rely on the mathematical derivation
       in the proof to determine the true answer value.
4. **Map the True Value to Our Choices:** Once you have the absolute
    true answer *value* (determined from either the stated key or the
    proof), find the corresponding option number in **our English
    `choices`**. This step is crucial to handle cases where the
    options were reordered during translation.

To make the best judgment, you must understand how the data was
    created and where errors can be introduced.

**CRITICAL: Understand the Data Pipeline to Find the Error Source:**
To identify the source of an error, you must first understand how the
    data was created. Here is the exact procedure we followed:

1. **PDF to Markdown Parsing:** We started with the original Persian
    exam PDFs and used an automated tool to parse them into markdown.
    This process sometimes introduces errors, like misinterpreting
    LaTeX (`\binom` as `\frac`) or failing to extract an image. The
    `persian_solution` field is the direct output of this step.
```
```

```
2. **LLM Translation:** The parsed Persian markdown was then
    translated into English using a Large Language Model. This step
    can introduce its own errors, especially with Right-to-Left (RTL)
    language nuances. For example, the order of items in a list (`7,
    10, 11`) might be incorrectly reversed (`11, 10, 7`). The
    `english_solution`, `problem`, and `choices` fields are the output
    of this step.
3. **Image Separation:** We manually separated images from the parsed
    text. It's possible an image was missed or mismatched during this
    step.


**Ground Truth:**
You have access to `crawled_persian_markdown`. This is the ultimate
    source of truth for what the official source published. However,
    the official source may omit the full solution: sometimes it
    provides only hints, and sometimes it includes only the problem
    with no solution. In such cases, downstream English content may
    come from a trusted alternative (e.g., official PDF extraction).
    Therefore:
- Use `crawled_persian_markdown` as the authoritative reference for
    the official problem statement and any content it does include.
- Absence of a solution in `crawled_persian_markdown` does NOT imply
    an error in the English solution by itself; In these cases, we
    have extracted the solution from the official PDF, which adds the
    possibility of mistakes in the english solution. Evaluate
    consistency using all provided references.

**Your Root Cause Analysis Procedure**

To accurately identify the error, you must follow this exact two-step
    procedure. Do not skip steps or classify an error until you have
    traced its origin according to this hierarchy of suspicion.

**Step 1: Verify Translation Fidelity (Check for Pipeline Errors)**
Your first and most important task is to meticulously compare the
    English text fields (`problem`, `context`, `choices`,
    `english_solution`) against the `crawled_persian_markdown` (the
    ground truth).
*   **Outcome:** If you find any discrepancya mistranslated equation,
    a reversed list, a sentence that doesn't matchthe root cause is a
    **Pipeline Error**. You must select the appropriate
    `Mistranslation...` or related category and set the `Pipeline
    Step` to `LLM Translation` or `PDF to Markdown Parsing`. **In this
    case, you must not proceed to Step 2.**

**Step 2: Analyze the Source (Check for Source Errors)**
If, and only if, you have confirmed that the English data is a
    faithful and accurate translation of the
    `crawled_persian_markdown`, should you then analyze the Persian
    source for internal flaws.
*   **Outcome:** If you find a demonstrable mathematical error, a
    typo, or a notational abuse *within the Persian source itself*,
    the root cause is an **Original Source Error**. You must select
    the `OriginalSourceError` category and set the `Pipeline Step` to
    `External Source`.

**Common Error Patterns Stemming from this Pipeline:**

*   **`MistranslationEquation`:** **(Cause: Step 1 or 2)**. A
    mathematical variable, expression, or equation was parsed
    incorrectly or went missing during PDF extraction (e.g., `\binom`
```

```
    became `\frac`) or was mistranslated by the LLM. Compare the
    English version to both Persian versions to pinpoint the source.
*   **`MistranslationOrderingRTL`:** **(Cause: Step 2)**. The order of
    items in a list, question, or choices was reversed or scrambled
    during the Persian-to-English translation. This is a classic RTL
    vs. LTR issue.
*   **`MistranslationAnswerKey`:** **(Cause: Step 2 & manual
    intervention)**. The original problem had an issue (e.g., the
    correct answer value was not in the choices). We may have manually
    added the correct value to the English `choices`, but the
    LLM-translated `english_solution` text might still incorrectly
    state that the answer isn't available.
*   **`ManualErrorIncorrectGuess`:** **(Cause: Manual intervention)**.
    The original Persian source marked the correct option with a '?'
    or it was ambiguous. A human manually filled in the
    `correct_option` and `answer_value`. **Analyze the solution's
    mathematical reasoning in the `crawled_persian_markdown`. If this
    logic contradicts the manually entered answer, this is the correct
    category.** This is the only situation that allows for the final
    answer to be programmatically changed.
*   **`MissingImage`:** **(Cause: Step 1 or 3)**. An image referenced
    in the text is missing. Compare the `english_solution` to the
    `crawled_persian_markdown` to see if an image reference is present
    in the source but absent in the final version.
*   **`ImageUnderstandingIssue`:** **(External Cause)**. The error is
    not in the text, but in the model's inability to correctly
    interpret an image's content. The text across all versions is
    likely consistent.
*   **`OriginalSourceError`:** **(External Cause)**. The logical flaw
    exists in the official source material itself. **To claim this
    category, you must provide a mathematical counter-example or proof
    demonstrating the error.** You cannot claim an error simply
    because the source is vague, concise, or contains an unproven
    claim (the benefit of the doubt always goes to the source). This
    category includes typos, abuse of notation (e.g., wrong indexing,
    undefined variables), or demonstrable mathematical mistakes in the
    proof.
*   **`NoDiscernibleError`:** **(Cause: Upstream Validator False
    Positive)**. A meticulous comparison of the `english_solution`,
    `persian_solution`, and `crawled_persian_markdown` shows they are
    all consistent and logically sound. The error is likely a false
    positive from the initial upstream validation workflow. Use this
    category if you can find no fault in the data.

**Your Task:**
1. Meticulously compare the three data versions
    (`crawled_persian_markdown`, `persian_solution`,
    `english_solution`) to trace where the error was introduced.
2. Enumerate all distinct issues you find (do not stop at the "most
    likely" one). For each issue:
   - Assign the exact category from the list below.
   - Write a detailed, plausible scenario that references the
       specific pipeline step that caused it.
   - Add a confidence tag: `High`, `Medium`, or `Low`.
   - Group repeated occurrences of the same category under a single
       issue entry, and list all occurrences with precise
       locations/snippets.
   - Rate the impact severity as `Critical`, `Major`, or `Minor`.
  Order the issues by severity (Critical  Major  Minor). There is no
      cap on the number of issues; include minor typos/notation
      issues as well.

**Input Data:**
```

```
- Crawled Persian Markdown (Source of Truth):
    {crawled_persian_markdown}
- Our Parsed Persian Markdown: {persian_solution}
- English Problem: {problem}
- Context: {context}
- English Choices: {choices}
- English Solution: {english_solution}
- SVG XMLs (if any):
  ```
  {svg_sources}
  ```

Note on SVGs: The SVG XML snippets are provided as auxiliary aids to
    clarify equations or diagram content. The equivalent rendered PNG
    images are already embedded in the problem/solution/context. Use
    SVGs only to improve understanding; do not output or modify them.

Note on Context: The `context` field contains introductory text or
    diagrams that are essential for understanding the problem but are
    not part of the formal question. Treat it as part of the overall
    problem definition.

**Output Instructions:**
For each distinct issue you identify, format your analysis using the
    following markdown structure. If you find multiple issues, repeat
    this block for each one, separated by a horizontal rule (`---`).
    List issues in descending order of severity.

**Category:** [Exact category name]
**Severity:** [Critical | Major | Minor]
**Confidence:** [High | Medium | Low]
**Pipeline Step:** [PDF to Markdown Parsing | LLM Translation | Image
    Separation | Manual Intervention | External Source]
**Explanation:** [Detailed plausible scenario of how/why this issue
    occurred]
**Occurrences:**
- [Document: crawled_persian_markdown | persian_solution |
    english_solution | choices | problem]  [location/snippet]  [what
    is wrong vs expected]
- [add more bullets for each occurrence]
```

B.2.2 IssueAggregator

IssueAggregator Prompt

```
**Role:** You are a lead forensic analyst responsible for
    synthesizing reports from multiple junior analysts. You have
    received several `IssueDetectionReport`s for the same problem.
    Your task is to review them all and produce one final,
    authoritative report.

### How to Determine the True Final Answer

Before classifying an error, you must determine the ground truth for
    the final answer by following this strict hierarchy. This is the
    most critical part of your analysis.

1. **Find the Stated Answer Key:** First, check the
    `crawled_persian_markdown` for an explicit statement of the
    correct option, like "Option X is correct" (` X ł ł`).
2. **The Stated Answer is the Target:**
```

```
   *  If an explicit option is stated, find its corresponding
      **value** from the Persian `choices` list. This value is the
      **intended correct answer (the ground truth)**.
   *  If the mathematical proof derives a different value, this
      indicates a **fixable flaw (e.g., a typo, calculation error, or
      encoding issue) within the proof**. Your task is to assume the
      stated answer is correct and identify the flaw in the proof.
3. **Use the Proof as the Fallback:**
   *  If and only if the Persian source is ambiguous (e.g., "Option ?
      is correct"), you must then rely on the mathematical derivation
      in the proof to determine the true answer value.
4. **Map the True Value to Our Choices:** Once you have the absolute
    true answer *value* (determined from either the stated key or the
    proof), find the corresponding option number in **our English
    `choices`**. This step is crucial to handle cases where the
    options were reordered during translation.

**CRITICAL: Understand the Data Pipeline to Evaluate the Reports:**
To make the best judgment, you must understand how the data was
    created and where errors can be introduced.

1. **PDF to Markdown Parsing:** We started with original Persian exam
    PDFs and used a tool to parse them into markdown
    (`persian_solution`). This step can cause LaTeX errors or miss
    images.
2. **LLM Translation:** The parsed markdown was then translated into
    English (`english_solution`, `problem`, etc.). This step can cause
    Right-to-Left (RTL) ordering issues or other mistranslations.
3. **Image Separation & JSON Formatting:** Manual steps that could
    also introduce errors.
4. **Ground Truth:** The `crawled_persian_markdown` reflects what the
    official source published. It may omit full solutions; sometimes
    only hints or only the problem are present. Treat it as
    authoritative for what it contains, but absence of a solution
    there does not, by itself, invalidate an English solution obtained
    from trusted official PDFs. In these cases, we have extracted the
    solution from the official PDF, which adds the possibility of
    mistakes in the english solution.

**Common Error Patterns Stemming from this Pipeline:**

*   `MistranslationEquation`: Caused by Step 1 or 2.
*   `MistranslationOrderingRTL`: Caused by Step 2.
*   `MistranslationAnswerKey`: Caused by Step 2 & manual fixes.
*   `ManualErrorIncorrectGuess`: **(Cause: Manual intervention)**. The
    original Persian source marked the correct option with a '?' or it
    was ambiguous. A human manually filled in the `correct_option` and
    `answer_value`. **Analyze the solution's mathematical reasoning in
    the `crawled_persian_markdown`. If this logic contradicts the
    manually entered answer, this is the correct category.** This is
    the only situation that allows for the final answer to be
    programmatically changed.
*   `MissingImage`: Caused by Step 1 or 3.
*   `ImageUnderstandingIssue`: External issue with the image
    understanding capability of the model.
*   `OriginalSourceError`: **(External Cause)**. The logical flaw
    exists in the official source material itself. **To claim this
    category, you must provide a mathematical counter-example or proof
    demonstrating the error.** You cannot claim an error simply
    because the source is vague, concise, or contains an unproven
    claim (the benefit of the doubt always goes to the source). This
    category includes typos, abuse of notation (e.g., wrong indexing,
```

```
    undefined variables), or demonstrable mathematical mistakes in the
    proof.
*   `NoDiscernibleError`: The upstream validation was likely a false
    positive.

**The Hierarchy of Suspicion: Your Guiding Principle**

Your primary goal as the lead analyst is to determine the true origin
    of any reported error. You must follow this hierarchy, assuming
    that errors are more likely to come from our automated processes
    than from the original source material.

1. **Highest Suspention: Our Pipeline (Extraction & Translation)**
   *  This is the most likely source of error. Before considering any
       other cause, you must first rule out errors from PDF parsing or
       LLM translation.
   *  **Evidence:** A discrepancy between the English fields
       (`problem`, `solution`, etc.) and the
       `crawled_persian_markdown`.
   *  **Your Action:** If a pipeline error is confirmed, it is the
       primary cause. The goal is to make our data consistent with the
       source.

2. **Medium Suspicion: Minor Flaws in the Source Solution**
   *  If, and only if, you have confirmed the English data is a
       faithful translation, then consider minor errors in the source
       solution itself.
   *  **Evidence:** The source proof contains typos, bad phrasing, or
       non-standard notation but is otherwise logically sound.
   *  **Your Action:** Acknowledge the minor source flaw. This can be
       fixed automatically.

3. **Lowest Suspicion: Flaws in the Source Problem Statement or Final
    Answer**
   *  This is extremely rare. Assume the original problem statement
       and stated final answer are correct unless there is
       overwhelming and unambiguous evidence of an error (e.g., a
       completely unintelligible typo).

**Handling Combined Errors:**
If you find evidence of both a minor source error AND a subsequent
    translation error, your final report must prioritize fixing the
    source concept first, then addressing the translation based on
    that corrected concept.

**Your Task:**
1. Review all provided detection reports below. Note the categories,
    explanations, and confidence scores from each analyst.
2. Aggregate ALL distinct issues across reports; do not stop at the
    most likely one.
3. For each aggregated issue, provide: Category; Severity \[Critical
    | Major | Minor\]; Confidence \[High | Medium | Low\]; Pipeline
    Step; and grouped Occurrences (per-location bullets).
4. Order issues by Severity (Critical  Major  Minor), then by
    Confidence.
5. Choose ONE overall `final_category` (the dominant issue for
    executive labeling) and list all remaining categories in
    `secondary_categories`.
6. Set control flags from the entire merged set of issues (not only
    from `final_category`).
7. Produce your final aggregated report as a markdown document. Do
    not propose removing any image references; image content is
    essential and must be preserved.
```

```
### Aggregation Rules
- Deduplicate same-category issues across reports and union their
    occurrences.
- Severity: take the highest severity reported for that issue across
    reports.
- Confidence: High if most reports are High and there are no strong
    conflicts; otherwise Medium; Low if evidence is conflicting or
    weak.
- Pipeline Step: choose the step best supported by evidence; if
    mixed, state the primary step and note alternates.
- **Prioritize Pipeline Errors in Conflict:** When reports conflict,
    apply the Hierarchy of Suspicion. If one analyst reports a
    `Mistranslation` and another reports an `OriginalSourceError` for
    the same discrepancy, the `Mistranslation` diagnosis takes
    precedence. Only classify the issue as an `OriginalSourceError` if
    there is shared evidence that the English text is a *faithful
    translation* of a flawed Persian source. When in doubt, default to
    the pipeline error.

**Detection Reports from Junior Analysts:**
{issue_reports_md}

**Problem Data for Reference:**
- Crawled Persian Markdown (Source of Truth):
    {crawled_persian_markdown}
- Our Parsed Persian Markdown: {persian_solution}
- English Problem: {problem}
- English Choices: {choices}
- English Solution: {english_solution}
- Correct Option: {correct_option}
- Answer Value: {answer_value}
- SVG XMLs (if any):
  ```
  {svg_sources}
  ```
- Context: {context}

Note on SVGs: The SVG XML snippets are provided as auxiliary aids to
    clarify equations or diagram content. The equivalent rendered PNG
    images are already embedded in the problem/solution/context. Use
    SVGs only to improve understanding; do not output or modify them.

Note on Context: The `context` field contains introductory text or
    diagrams that are essential for understanding the problem but are
    not part of the formal question. Treat it as part of the overall
    problem definition.

Return only a single valid JSON object conforming to the schema
    below. Do not include any extra text or code fences. Keys must be
    double-quoted.

### Rules for Setting Control Flags
Your primary task is to review ALL detected issues from the junior
    analysts' reports and set the following boolean flags based on the
    *entire set* of findings. The `final_category` is for descriptive
    purposes only; these flags control the workflow.

1. **`is_original_source_error`**:
   - MUST be `true` if `OriginalSourceError` is present in ANY of the
      detected issues (either as a primary or secondary finding).
   - MUST be `false` otherwise.
```

```
2. **`is_image_understanding_issue`**:
   - MUST be `true` if `ImageUnderstandingIssue` OR `MissingImage` is
       present in ANY of the detected issues.
   - MUST be `false` otherwise.

3. **`requires_human_intervention`**:
   - MUST be `true` if `is_original_source_error` is `true` OR
       `is_image_understanding_issue` is `true`.
   - MUST be `false` otherwise.

**JSON Schema for Output:**

{
  "title": "AggregatedIssueReport",
  "type": "object",
  "properties": {
    "final_category": {
      "type": "string",
      "enum": [
        "MistranslationEquation",
        "MistranslationOrderingRTL",
        "MistranslationAnswerKey",
        "ManualErrorIncorrectGuess",
        "MissingImage",
        "ImageUnderstandingIssue",
        "OriginalSourceError",
        "NoDiscernibleError",
        "Other"
      ]
    },
    "requires_human_intervention": { "type": "boolean" },
    "is_original_source_error": {
      "type": "boolean",
      "description": "True if 'OriginalSourceError' appears in ANY
          detected issues (primary or secondary)."
    },
    "is_image_understanding_issue": {
      "type": "boolean",
      "description": "True if 'ImageUnderstandingIssue' or
          'MissingImage' was detected. Controls the workflow branch."
    },
    "secondary_categories": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": [
          "MistranslationEquation",
          "MistranslationOrderingRTL",
          "MistranslationAnswerKey",
          "ManualErrorIncorrectGuess",
          "MissingImage",
          "ImageUnderstandingIssue",
          "OriginalSourceError",
          "NoDiscernibleError",
          "Other"
        ]
      }
    },
    "plausible_scenario_md": { "type": "string" },
    "aggregated_report_md": { "type": "string" }
  },
```

```
  "required": ["final_category", "requires_human_intervention",
      "is_original_source_error", "is_image_understanding_issue",
      "plausible_scenario_md", "aggregated_report_md"]
}


### Output Structure for `aggregated_report_md`
- Header: Final Category + Flags (concise, visible summary).
- Issues Breakdown: one block per issue with Category, Severity,
    Confidence, Pipeline Step, and grouped Occurrences (per-location
    bullets).
- Evidence Synthesis: explain how reports were merged, how conflicts
    were resolved, and why the chosen pipeline step/labels were
    selected.
- Final Decision & Rationale: why this `final_category` dominates;
    how flags were computed from the whole set.

**Example Output:**

**Final Category:** MistranslationEquation
**Requires Human Intervention:** false

## Issues Breakdown
### Issue 1
- **Category:** MistranslationEquation
- **Severity:** Major
- **Confidence:** High
- **Pipeline Step:** LLM Translation
- **Occurrences:**
  - Document: english_solution  snippet shows `\\frac{n}{k}`; expected
      `\\binom{n}{k}`
  - Document: problem  heading formula mirrored incorrectly

### Issue 2
- **Category:** MistranslationOrderingRTL
- **Severity:** Minor
- **Confidence:** Medium
- **Pipeline Step:** LLM Translation
- **Occurrences:**
  - Document: choices  order reversed (11, 10, 7 vs 7, 10, 11)

## Evidence Synthesis
Reports 1 and 3 independently confirm equation mistranslation with
    high confidence; Report 2 identifies the ordering issue. We merge
    same-category findings and union occurrences. Severity is taken as
    the highest reported; confidence is High for Issue 1 due to
    consistent evidence, Medium for Issue 2 due to partial agreement.

## Final Decision & Rationale
The dominant issue is MistranslationEquation (Major, High), thus it
    is selected as `final_category`. MistranslationOrderingRTL is
    retained via `secondary_categories`. Control flags are computed
    from the entire set of issues.
```

B.2.3 SOLUTIONENGAGER

SolutionEngager Prompt

```
**Role:** You are an expert mathematician tasked with expanding a
    very concise mathematical solution into a complete, rigorous
    proof. Your goal is to fill in all omitted steps and justify every
```

```
    claim. During this process, if you encounter any statement that
    you can definitively prove is incorrect, document it as an error.

**Understanding Our Data Pipeline and Why This Task Matters**
To perform this role correctly, you must understand how our data was
    created and why errors might exist:

1. **Original Source:** We started with official Persian exam PDFs
    from math olympiads and used automated tools to parse them into
    markdown. This parsing can introduce errors like misinterpreting
    LaTeX ('\binom' as '\frac') or missing images.

2. **Translation Pipeline:** The parsed Persian markdown was then
    translated into English using an LLM. This can introduce
    translation errors, especially with Right-to-Left language issues
    (e.g., reversing the order of items in lists).

3. **Manual Processing:** Images were separated manually, and
    everything was formatted into JSON for our database.

4. **Current Situation:** Our validation workflow has flagged this
    problem as potentially containing an error. However, we suspect
    the error might be in the original source material itselfeither a
    typo, unclear phrasing, or an actual mathematical mistake made
    under deadline pressure.

**Your Critical Role in This Pipeline:**
The upstream validation detected an issue, but it's unclear whether
    this is due to:
- A real mathematical error in the original source
- Poor/unclear phrasing that makes a correct solution seem wrong
- Translation/processing errors from our pipeline

Since the original solutions are extremely concise (typical of
    olympiad publications), directly analyzing them often leads to
    false positivesa statement might seem wrong simply because its
    justification was omitted. Your job is to expand the solution
    completely, and during this process, determine if any claims are
    genuinely mathematically incorrect.

### Your Primary Directive: The Hierarchy of Truth

Before you begin your analysis, you must understand the ground truth
    of the problem. Your entire analysis must be based on the
    following strict hierarchy.

1. **Find the Stated Answer Key:** First, check the
    `crawled_persian_markdown` for an explicit statement of the
    correct option, like "Option X is correct" (` X ł ł`).

2. **The Stated Answer is the Target:**
   *  If an explicit option is stated, find its corresponding
      **value** from the Persian `choices` list. This value is the
      **intended correct answer (the ground truth)**. Your job is to
      treat this answer as correct.
   *  If the mathematical proof in the solution appears to derive a
      different value, this signals a **flaw within the proof**. Your
      task is not to challenge the answer, but to expand the proof
      and pinpoint the exact typo, calculation error, or logical leap
      that causes it to deviate from the correct target answer.

3. **Use the Proof as the Ground Truth (Fallback Case):**
```

```
  *  If, and only if, the Persian source is ambiguous (e.g., states
     "Option ? is correct"), does the burden of proof shift. In this
     specific case, you must then rely on the mathematical
     derivation in the proof to determine the true answer.

**CRITICAL PRINCIPLE: Benefit of the Doubt**
You must give the original solution the benefit of the doubt. Only
    flag something as an error if you can provide concrete evidence
    (counterexample, derivation, proof, or clear reasoning) that
    demonstrates the statement is mathematically incorrect. You cannot
    flag something as wrong simply because it lacks justification or
    seems unclear.

**Source Material Selection**
Follow this decision recipe, in order:
1. **Persian has hints + solution:** Use both together. Expand the
    solution while leveraging the hints for structure and intent.
2. **Persian has solution only (concise):** Expand that Persian
    solution into a complete, rigorous proof.
3. **Persian has hints; English has solution:** Combine them. Use
    Persian hints to guide structure and intent, and fill in the
    detailed steps from the English solution. If there is a conflict,
    prefer the Persian sources intent and notation. Explicitly
    annotate any conflicts and explain how English steps were adapted
    to align with the Persian intent/notation.
4. **Persian has neither solution nor hints:** Use the English
    solution as the fallback source.

Notation Policy: Preserve the original (Persian) notation when it is
    nonstandard but internally consistent. Define symbols upon first
    use and, if helpful, include a parenthetical mapping to standard
    notation. Do not silently normalize unless absolutely necessary;
    prefer preserving fidelity and explaining.

**Your Task:**
Engage honestly with each claim. When uncertain about a claims
    correctness, assume it is correct and attempt to justify it. If,
    during justification, you become confident it is incorrect,
    explain mathematically why (proof or counterexample). Aim for full
    rigor; include all necessary steps. Prefer clear and complete
    reasoning over brevity.

1. **Expand the Solution:** Rewrite the solution fully and clearly,
    providing justification for each claim. For every claim, either
    confirm its correctness with reasoning, orif you are confident it
    is wrongprovide a mathematical refutation (proof or
    counterexample).
2. **Document Proven Errors:** If during expansion you encounter a
    statement that you can prove is incorrect, document it with
    concrete evidence.
3. **Assess Overall Integrity:** Determine if the original solution's
    core logic is sound or fundamentally flawed.
4. **Reconcile OriginalSource vs Pipeline Errors:** If your expansion
    shows the source is correct and prior issues came from
    parsing/translation/formatting, explicitly state this downgrade.
    If issues are typos/notation/wording, treat them as Minor, Fixable
    (not an originalsource error). Only assert a true
    OriginalSourceError when you can exhibit a concrete mathematical
    contradiction or an unfixable flaw in the core reasoning.

### Final Assessment Criteria
```

```
Your final assessment is critical for the next stage of the workflow.
    Use the following definitions to make your judgment:

**Choose "Major Logical Flaw" IF:**
- The core method or theorem used in the proof is fundamentally
    incorrect and could not lead to the correct answer, even with
    minor fixes.
- The proof contains a chain of incorrect logical steps that makes
    the entire argument unsalvageable.
- Fixing the proof would require a complete rewrite using a different
    mathematical approach, not just a series of simple corrections.

**Choose "Minor, Fixable Issue" IF:**
- The overall method of the proof is sound, but it contains localized
    errors such as typos, calculation mistakes, incorrect variable
    names, or notational errors.
- The proof correctly reaches the stated answer key, but you
    identified a specific flaw in a few steps that needs correction.
- The logic is correct but is presented in a very vague or confusing
    way that can be clarified with minor rewriting.

**Inputs:**
- **Initial Issue Report:** {aggregated_report_md}
- **Persian Source:** {crawled_persian_markdown}
- **English Source:** {english_solution}
- **Problem Context:** {problem}
- **Choices:** {choices}
- **Correct Option:** {correct_option}
- **Answer Value:** {answer_value}
- **SVG XMLs:** {svg_sources}
- **Context:** {context}

Note on SVGs: The SVG XML snippets are provided as auxiliary aids to
    clarify equations or diagram content. The equivalent rendered PNG
    images are already embedded in the problem/solution/context. Use
    SVGs only to improve understanding; do not output or modify them.

Note on Context: The `context` field contains introductory text or
    diagrams that are essential for understanding the problem but are
    not part of the formal question. Treat it as part of the overall
    problem definition.

**Output Format:**

## Source Analysis
(State which source you used and whether it contained a complete
    solution)

## Expanded Rigorous Solution
(Your complete, step-by-step expansion of the original solution)

## Claim-by-Claim Justification
For each claim referenced in the original solution (and any newly
    clarified intermediate claim), provide:
- **Claim:** [quote or precise paraphrase]
- **Status:** [Confirmed | Uncertain-but-plausible |
    Incorrect-with-proof]
- **Justification/Evidence:**
  - If Confirmed or Uncertain-but-plausible: brief reasoning or
      derivation showing why it holds or why it is plausibly correct.
  - If Incorrect-with-proof: a concise derivation or counterexample
      demonstrating the error; citing well-known theorems with brief
      justification is acceptable.
```

```
 - **Initial Correction Proposal (if applicable):** If this claim can
     be corrected with a minor, surgical edit (e.g., typo, index,
     notation, single-sentence clarification), propose the precise
     minimal change while preserving images and structure. If it
     appears to require structural changes, note that no minor
     proposal is appropriate here.

## Holistic Fixability Assessment
Provide a holistic judgment of fixability across all claims taken
    together. Label and justify:
- **Overall Fixability:** [Minor-surgical | Major-rewrite | Unknown]
- **Narrative:** Explain how the errors were introduced (e.g.,
    translation pipeline, parsing, formatting) and whether a
    straightforward, coherent set of minimal edits can resolve all
    issues. Consider the solution as a whole: if a clear narrative and
    concise set of targeted edits suffice, it is Minor-surgical; if
    the approach/method is invalid or requires a substantial rewrite,
    it is Major-rewrite.

## Documented Errors (if any)
(Any statements you can prove are incorrect, with concrete evidence.
    Provide a concise derivation or counterexample; citing well-known
    theorems with brief justification is acceptable. **Remember: if
    the proof derives an answer that contradicts the stated answer
    key, the error is in the proof, not the answer key.** IMPORTANT:
    Reference the specific location in the ORIGINAL source material
    where each error occurs, not your expanded version.)

## Final Assessment
(Either "Minor, Fixable Issue" or "Major Logical Flaw")

## Proposed Corrections Summary (if Minor/Fixable)
Consolidate all minor, surgical proposals into a coherent, minimal
    set of edits that resolves the issues. Do not delete images;
    preserve original notation unless you define a clear mapping.
```

B.2.4 IssueDetectorWithEngagement

IssueDetectorWithEngagement Prompt

```
**Role:** You are a senior decision-maker in an AI data pipeline.
    Your task is to synthesize a deep-dive analysis of a math problem
    and determine if the identified source error requires human
    intervention or can be fixed automatically.

### How to Determine the True Final Answer

Before making your final decision, you must re-verify the ground
    truth for the final answer by following this strict hierarchy.

1. **Find the Stated Answer Key:** First, check the
    `crawled_persian_markdown` for an explicit statement of the
    correct option, like "Option X is correct" (` X ł ł`).
2. **The Stated Answer is the Target:**
   *  If an explicit option is stated, find its corresponding
      **value** from the Persian `choices` list. This value is the
      **intended correct answer (the ground truth)**.
   *  If the mathematical proof derives a different value, this
      indicates a **fixable flaw (e.g., a typo, calculation error, or
      encoding issue) within the proof**. Your task is to assume the
      stated answer is correct and identify the flaw in the proof.
```

```
3. **Use the Proof as the Fallback:**
   *  If and only if the Persian source is ambiguous (e.g., "Option ?
       is correct"), you must then rely on the mathematical derivation
       in the proof to determine the true answer value.
4. **Map the True Value to Our Choices:** Once you have the absolute
    true answer *value* (determined from either the stated key or the
    proof), find the corresponding option number in **our English
    'choices'**. This step is crucial to handle cases where the
    options were reordered during translation.

### Understanding the Context
A previous stage ('SolutionEngager') has performed a detailed,
    evidence-based analysis of the problem's solution. Your job is to
    use that analysis, combined with your knowledge of our data
    pipeline, to make the final call.

**Common Error Patterns:**
*  'ManualErrorIncorrectGuess': A human's guess for the answer was
    contradicted by the source proof.
*  'OriginalSourceError': The source material itself contains a
    demonstrable mathematical mistake, typo, or notational error.
*  'Mistranslation...': An error was introduced during translation.

### How to Interpret the Engagement Analysis

The 'SolutionEngager' uses the following strict criteria to make its
    assessment. You must use these same definitions to interpret its
    findings.

**"Major Logical Flaw" means:**
- The core method or theorem used in the proof is fundamentally
    incorrect and could not lead to the correct answer, even with
    minor fixes.
- The proof contains a chain of incorrect logical steps that makes
    the entire argument unsalvageable.
- Fixing the proof would require a complete rewrite using a different
    mathematical approach.

**"Minor, Fixable Issue" means:**
- The overall method of the proof is sound, but it contains localized
    errors such as typos, calculation mistakes, incorrect variable
    names, or notational errors.
- The logic is correct but is presented in a vague or confusing way
    that can be clarified with minor rewriting.

**Your Decision Criteria:**
Based on the 'Detailed Engagement Analysis' and the full context, you
    must decide:

**Requires Human Intervention ('true') IF:**
- The engagement analysis proves a **Major Logical Flaw** in the
    source material's core reasoning that cannot be salvaged by a
    small number of targeted edits.
- The errors are so complex or numerous that they require domain
    expertise beyond the scope of an automated fix plan.

**Can Be Handled Automatically ('false') IF:**
- The engagement analysis shows a coherent, straightforward narrative
    of introduced errors (e.g., translation/parsing/formatting) and a
    concise, minimal set of targeted edits can resolve all issues
    (Minor, Fixable). The core logic is sound.
- The analysis confirms a 'ManualErrorIncorrectGuess' where the
    correct answer can be reliably derived from the source proof.
```

```
**CRITICAL PRINCIPLE:** Trust the evidence-based assessment. Major vs
    Minor is about repair scope (structural rewrite vs surgical
    edits), not just about whether an error is proven. If the
    `SolutionEngager` could not mathematically prove an error, give
    the benefit of the doubt to the source and classify the issue as
    fixable.

### Post-Engagement Reconciliation: Re-applying the Hierarchy of
    Suspicion

The deep-dive analysis provides you with powerful new evidence. Your
    primary task is to use this evidence to re-apply the Hierarchy of
    Suspicion and confirm or overturn the initial
    `OriginalSourceError` diagnosis.

1. **Re-check for Pipeline Errors:** The `SolutionEngager` may have
    uncovered subtle translation or parsing artifacts that were not
    obvious before. For example, a confusing sentence in the source
    might have been mistranslated, making it seem like a logical error
    when it was not.
   *  **Action:** If the engagement report provides strong evidence
      that the issue is actually a **Pipeline Error**
      (mistranslation, parsing), you must treat the issue as fixable.

2. **Re-assess the Source Error:** If the engagement confirms the
    English text is a faithful translation, now re-evaluate the source
    flaw based on its severity.
   *  **Is it a Minor Flaw?** The engagement may have proven the
      error is just a typo, a notational inconsistency, or a poorly
      phrased sentence, while the core logic remains sound. This is a
      "Minor, Fixable Issue".
   *  **Is it a Major Flaw?** The engagement may have provided a
      mathematical proof that the source's core reasoning is
      unsalvageable. This is a "Major Logical Flaw".

Your final decision on `requires_human_intervention` must be based on
    this re-evaluation. Downgrading a supposed `OriginalSourceError`
    to a fixable pipeline or minor source error is a primary goal of
    this stage.

**Inputs:**
- **Initial Issue Report:** {aggregated_report_md}
- **Detailed Engagement Analysis:** {solution_engagement_report_md}
- **Persian Source:** {crawled_persian_markdown}
- **English Source:** {english_solution}
- **Problem Context:** {problem}
- **Choices:** {choices}
- **Correct Option:** {correct_option}
- **Answer Value:** {answer_value}
- **SVG XMLs:** {svg_sources}
- **Context:** {context}

Note on Context: The `context` field contains introductory text or
    diagrams that are essential for understanding the problem but are
    not part of the formal question. Treat it as part of the overall
    problem definition.

**JSON Schema:**
{
  "title": "SourceIssueClassification",
  "type": "object",
  "properties": {
```

```
    "requires_human_intervention": {
      "type": "boolean",
      "description": "True if the issue requires human review, false if
          it can be handled automatically"
    },
    "reasoning": {
      "type": "string",
      "description": "Brief justification for the decision, explaining
          why the issue is deemed major or minor based on the new,
          comprehensive context."
    }
  },
  "required": ["requires_human_intervention", "reasoning"]
}
```


B.2.5 EngagementReportSynthesizer

EngagementReportSynthesizer Prompt

```
**Role:** You are the **Lead Analyst** in a multi-stage AI workflow
    designed to automatically detect and repair errors in math
    problems. You are the crucial synthesis point in the most complex
    branch of the workflow.

**The Big Picture: What We Are Doing**
Our overall goal is to create a reliable, automated system that can
    fix complex issues in our dataset. Think of it as an assembly line
    of AI specialists. An early specialist (`IssueAggregator`) has
    flagged a problem with a potentially critical
    `OriginalSourceError`.

Because this is a serious accusation, the workflow paused the normal
    "fix-it" process and instead launched a deep-dive forensic
    investigation. Two expert agents were dispatched:
1. `SolutionEngager`: This agent performed a detailed, step-by-step
    logical breakdown of the original Persian solution to understand
    its core reasoning.
2. `IssueDetectorWithEngagement`: This agent used the
    `SolutionEngager`'s report to make a final, expert judgment on the
    nature and fixability of the source error.

**Your Specific Role in this Workflow**
You are the specialist who receives the initial, high-level alert
    (`aggregated_report_md`) and the detailed reports from the
    forensic investigation (`solution_engagement_report_md` and
    `source_issue_classification_md`).

Your mission is to **create the single, final, and authoritative
    `AggregatedIssueReport` JSON object**. The next agent in the
    pipeline, the `FixPlanner`, will base its entire repair strategy
    on the report you generate. The quality and coherence of your
    output will determine whether the problem is fixed correctly or
    the entire process fails.

**Your Task:**

Your mission is to produce the final, authoritative
    `AggregatedIssueReport` JSON object. To do this, you must
    synthesize all inputs by narrating the outcome of the
    post-engagement re-evaluation, guided by the Hierarchy of
    Suspicion.
```

```
1. **Establish the Baseline:** Start with the `Initial Report`. Note
   its original `final_category` and findings.
2. **Apply the Hierarchy of Suspicion Lens:** Use the detailed
   evidence from the `Engagement Report` and `Final Classification`
   to re-evaluate the baseline findings.
   *  Did the engagement reveal a **Pipeline Error**
      (mistranslation/parsing) that was previously misdiagnosed as a
      source error?
   *  If not, did the engagement confirm a source error but classify
      it as **Minor and Fixable** (e.g., typo, notational issue)
      rather than a Major Logical Flaw?
3. **Synthesize the Narrative:** In the `plausible_scenario_md` and
   `aggregated_report_md`, you must tell the story of this
   re-evaluation. For example: "Initially, the issue was flagged as
   an OriginalSourceError. However, a deep-dive analysis revealed
   that the confusing sentence in the English solution was actually a
   mistranslation of a complex but correct statement in the Persian
   source. Therefore, the issue has been downgraded to a
   MistranslationEquation."
4. **Update Categories and Flags:** Based on your new understanding,
   determine the final, correct `final_category` and
   `secondary_categories`. Critically, you must re-compute all
   boolean flags (`requires_human_intervention`,
   `is_original_source_error`, etc.) based on this *final* set of
   issues, following the Decision Standard below.
5. **Generate the Final Report:** Ensure the `aggregated_report_md`
   contains all required sections (Issues Breakdown, Evidence
   Synthesis, Final Decision, Change Log, etc.) reflecting your
   synthesized findings.

**Inputs:**

1. **Initial Report (`aggregated_report_md`):**
   {aggregated_report_md}

2. **Engagement Report (`solution_engagement_report_md`):**
   {solution_engagement_report_md}

3. **Final Classification (`source_issue_classification_md`):**
   {source_issue_classification_md}
   (Formatted markdown produced by `FormatSourceIssueClassification`.)

4. **Problem Data for Reference:**
   - Crawled Persian Markdown (Source of Truth):
      {crawled_persian_markdown}
   - English Problem: {problem}
   - English Choices: {choices}
   - English Solution: {english_solution}
   - Correct Option: {correct_option}
   - Answer Value: {answer_value}
   - SVG XMLs (if any):

     {svg_sources}
   - Context: {context}

Note on Context: The `context` field contains introductory text or
   diagrams that are essential for understanding the problem but are
   not part of the formal question. Treat it as part of the overall
   problem definition.

### Decision Standard for Human Intervention (Post-Engagement)
```

```
You must set the final `requires_human_intervention` flag based on
    the outcome of your re-evaluation using the Hierarchy of Suspicion:

-  Set to `true` ONLY if the engagement confirms a **Major Logical
    Flaw** in the source's core reasoning that is not salvageable by
    minor edits, OR if an image issue blocks repair.
-  Set to `false` if the re-evaluation downgrades the issue to a
    **Pipeline Error** OR a **Minor, Fixable Source Error**.

**Output Instructions:**
Produce a single, valid JSON object with double-quoted keys that
    conforms strictly to the `AggregatedIssueReport` schema provided
    below. Do NOT add any extra text, markdown, explanations, or code
    fences. Return only the JSON object.

**JSON Schema for Output:**
```json
{
 "title": "AggregatedIssueReport",
 "type": "object",
 "properties": {
  "final_category": {
   "type": "string",
   "enum": [
    "MistranslationEquation",
    "MistranslationOrderingRTL",
    "MistranslationAnswerKey",
    "ManualErrorIncorrectGuess",
    "MissingImage",
    "ImageUnderstandingIssue",
    "OriginalSourceError",
    "NoDiscernibleError",
    "Other"
   ]
  },
  "requires_human_intervention": { "type": "boolean" },
  "is_original_source_error": {
   "type": "boolean",
   "description": "True if 'OriginalSourceError' was detected among
       any of the issues. Controls the workflow branch."
  },
  "is_image_understanding_issue": {
   "type": "boolean",
   "description": "True if 'ImageUnderstandingIssue' or
       'MissingImage' was detected. Controls the workflow branch."
  },
  "secondary_categories": {
   "type": "array",
   "items": { "type": "string" }
  },
  "plausible_scenario_md": { "type": "string" },
  "aggregated_report_md": { "type": "string" }
 },
 "required": ["final_category", "requires_human_intervention",
     "is_original_source_error", "is_image_understanding_issue",
     "plausible_scenario_md", "aggregated_report_md"]
}
```
```

B.2.6 FIXPLANNER

FixPlanner Prompt

```
**Role:** You are an expert AI data repair specialist. Your task is
    to analyze an issue report and the corresponding problem data,
    then create a clear, step-by-step markdown plan to fix the data.

**How to Interpret the Issue Report: The Hierarchy of Suspicion**

Before you create a single instruction, you must understand the
    origin of the error as determined by the `Aggregated Issue
    Report`. Your plan must be tailored to the error's source,
    following this hierarchy:

1. **If the error is from our Pipeline (Extraction/Translation):**
   *  **Your Goal:** Make our data a perfect reflection of the
       `crawled_persian_markdown` source.
   *  **Your Plan:** Create instructions to correct mistranslations,
       fix parsing errors, and align our data with the ground truth.

2. **If the error is a Minor Flaw in the Source Solution:**
   *  **Your Goal:** Correct the minor flaw (e.g., typo, notational
       error) in the source's logic and reflect that fix in our
       English data.
   *  **Your Plan:** Your instructions should surgically correct the
       `english_solution_local_images` to fix the issue.

3. **If there are Combined Errors (Source + Pipeline):**
   *  **Your Goal:** Create a plan that addresses the root cause
       first.
   *  **Your Plan:** Your instructions must be ordered correctly.
       First, an instruction to address the conceptual fix needed for
       the source error. Second, an instruction to fix the translation
       based on that now-corrected concept.

4. **If the `Aggregated Issue Report`'s `final_category` is
    `NoDiscernibleError` and there are no `secondary_categories`:**
   *  **Your Goal:** Confirm that no changes are needed and produce a
       plan stating this explicitly.
   *  **Your Plan:** You must generate a plan containing a single
       instruction: "No discernible error was found. The data is
       correct as-is and requires no changes."

### CRITICAL RULES FOR PLANNING FIXES

Your authority to make changes is strictly limited. While your
    primary goal is to create a complete plan to fix all issues in the
    report, you must operate within the following non-negotiable
    constraints:

#### RULE 0: CONFLICT RESOLUTION
Your primary goal is to follow all rules. If you find that fixing an
    issue according to one rule (e.g., `RULE 3`) would force you to
    violate another rule (e.g., `RULE 1`), you must prioritize safety.
    Your plan should:
1. Perform any minor, safe fixes that do not cause a conflict.
2. Clearly state the nature of the rule conflict you encountered
    (e.g., "Correcting the solution to match the updated problem would
    require a full rewrite, which violates RULE 1.").
3. Explicitly recommend that the problem requires human intervention.

#### RULE 1: MODIFICATIONS MUST BE MINOR AND SURGICAL
```

```
You are **forbidden** from rewriting entire solutions. The goal is to
    repair, not replace.
*  **You CAN:** Make minor edits like correcting typos, changing
    variables, fixing indices, or modifying equations within a
    sentence. You may rewrite one or two sentences if absolutely
    necessary to correct a specific, localized error.
*  **You CANNOT:** Propose a total rewrite, restructure the entire
    logical flow, or add large new paragraphs of explanation.

#### RULE 2: THE FINAL ANSWER IS SACROSANCT

Your plan must be generated by following this exact procedure for
    handling the final answer.

**Step 1: Determine if the Database Answer is Correct**
Your first job is to determine the absolute true answer by applying
    the official hierarchy to the `crawled_persian_markdown`.
-  If the source states an explicit answer (e.g., "Option 3 is
    correct"), that is the ground truth.
-  If the source is ambiguous (e.g., "Option ?"), then the answer is
    the one derived from the proof.

**Step 2: Plan the Fix Based on the Issue Category**
You are **strictly forbidden** from planning any changes to
    `correct_option` or `answer_value` unless the issue category is
    `ManualErrorIncorrectGuess`.
-  **IF the category is `ManualErrorIncorrectGuess`:** Your plan must
    update the database `correct_option` and `answer_value` to match
    the ground truth you derived in Step 1.
-  **IF the issue is a flaw in the proof** (i.e., the proof's result
    does not match the stated answer key): Your plan must focus on
    making a **minor, surgical correction** to the proof text in
    `english_solution_local_images` so that it correctly leads to the
    stated ground truth answer. **Do not change the answer itself.**
-  **IF the issue is anything else** (e.g., `OriginalSourceError`,
    `MistranslationAnswerKey`): Your plan must only address textual
    issues and **must not** alter `correct_option` or `answer_value`.

#### RULE 3: UPHOLD THE HIERARCHY OF TRUTH

Your primary directive is to ensure the data is a high-fidelity
    representation of the original Persian source
    (`crawled_persian_markdown`). All fixes must follow this strict
    hierarchy, where lower-priority data is always corrected to match
    higher-priority data.

1. **Ultimate Authority (`crawled_persian_markdown`):** This is the
    absolute ground truth.
2. **Problem Definition (`problem`, `context`, `choices`):** These
    fields must be a faithful translation of the Ultimate Authority.
3. **Derived Explanation (`english_solution_local_images`):** This
    field must correctly solve the problem as defined in the `problem`
    field.

- **You MUST:** If the `context` contains a typo or mistranslation
    (when compared to the Ultimate Authority), your plan must correct
    the `context` field.
- **You MUST:** If the `problem` has a typo or mistranslation (when
    compared to the Ultimate Authority), your plan must correct the
    `problem` field AND then also correct the
    `english_solution_local_images` so it solves the now-correct
    problem.
```

```
- **You MUST NOT:** Ever "fix" the `problem` field to justify an
    error in the `english_solution_local_images`. The solution always
    yields to the problem.

**Inputs:**

1. **Aggregated Issue Report (`aggregated_report_md`):** This is the
    ground truth. It describes what is wrong with the problem.

**Your Goal:**
Generate a list of clear, actionable instructions describing the
    complete, cascading changes required. Your plan must be
    exhaustive; every distinct issue mentioned in the Aggregated Issue
    Report, regardless of whether it is the `final_category` or a
    `secondary_category`, must have a corresponding step in your plan.
    Focus only on minimal edits.

**Constraints (Critical):**
- Do not propose removing, renaming, or altering any image
    references. Image content is essential and must be preserved.
- If an instruction would implicitly remove an image (e.g., replacing
    a section that contains images), rewrite the instruction to keep
    the images intact and only change the necessary text.
- Never instruct to delete image markdown (e.g., lines that start
    with `![](` or similar). Images must remain present in the final
    content.

**Examples of Good Fix Plans:**

**Simple Example (Single Issue):**
*  **Scenario:** The report indicates that the `correct_option` is 3,
    but the logic clearly points to the answer value found in option 5.
*  **Good Plan:**
   1. **Instruction:** The `correct_option` field is incorrect. It
       should be changed from 3 to 5.
      *  **Target Fields:** `correct_option`
      *  **Rationale:** The issue report identifies this as an error,
          and the solution's logic derives the answer found in option
          5.
   2. **Instruction:** Update the `answer_value` field to match the
       content of option 5.
      *  **Target Fields:** `answer_value`
      *  **Rationale:** This is a cascading change to keep the answer
          value consistent with the corrected option.

**Complex Example (Multiple Issues):**
*  **Scenario:** The report's main issue is
    `ManualErrorIncorrectGuess` (the `correct_option` is wrong) but it
    also notes a minor typo in the last sentence of the solution.
*  **Good Plan:**
   1. **Instruction:** The `correct_option` field is incorrect. It
       should be changed from 2 to 4.
      *  **Target Fields:** `correct_option`
      *  **Rationale:** The issue report identifies this as a Manual
          Error, and the solution's logic derives the answer found in
          option 4.
   2. **Instruction:** Update the `answer_value` field to match the
       content of option 4.
      *  **Target Fields:** `answer_value`
      *  **Rationale:** This is a cascading change to keep the answer
          value consistent with the corrected option.
```

```
   3. **Instruction:** In the 'english_solution_local_images',
       correct a typo in the last sentence. Change "teh final anser"
       to "the final answer".
       *  **Target Fields:** 'english_solution_local_images'
       *  **Rationale:** The report noted a secondary typo issue that
           needs to be addressed for clarity.

**"No-Op" Example (No Error Found):**
*  **Scenario:** The report's 'final_category' is
    'NoDiscernibleError' and 'secondary_categories' is empty.
*  **Good Plan:**
   1. **Instruction:** No discernible error was found. The data is
       correct as-is and requires no changes.
       *  **Target Fields:** 'None'
       *  **Rationale:** The Aggregated Issue Report concluded that
           the initial validation was a false positive and the data is
           correct.

**Aggregated Issue Report:**
{aggregated_report_md}

**Text Fields to Analyze:**
- problem: {problem}
- choices: {choices}
- english_solution_local_images: {english_solution}
- context: {context}
- correct_option: {correct_option}
- answer_value: {answer_value}
 - SVG XMLs (if any):
   '''
   {svg_sources}
   '''

Note on SVGs: The SVG XML snippets are auxiliary. The equivalent PNG
    renderings are already present in the context. Use SVGs only to
    disambiguate equations or figure details when forming the plan; do
    not propose editing or outputting SVGs.

Generate your 'FixPlan' as a markdown document.

**Required Output Structure:**

You must generate a markdown document with a level 3 header '### Fix
    Plan' and a numbered list of instructions. Each instruction must
    contain a nested list with the 'Target Fields' and 'Rationale'.

'''markdown
### Fix Plan

1. **Instruction:** [A clear, natural language instruction describing
    the complete change.]
   *  **Target Fields:** [A comma-separated list of field names,
       e.g., 'correct_option', 'answer_value']
   *  **Rationale:** [A brief explanation for why this fix is
       necessary.]
2. **Instruction:** [The next instruction, if any.]
   *  **Target Fields:** [...]
   *  **Rationale:** [...]
'''

**Example Output:**
'''markdown
### Fix Plan
```

````
1. **Instruction:** The `correct_option` field is incorrect. It
    should be changed from 3 to 5.
   *  **Target Fields:** `correct_option`
   *  **Rationale:** The aggregated report indicates that while the
       solution logic is sound, it points to the answer value
       contained in option 5, not option 3.
2. **Instruction:** Update the `answer_value` field to match the
    numerical value or content of the new correct option (option 5).
   *  **Target Fields:** `answer_value`
   *  **Rationale:** This is a cascading change required to keep the
       `answer_value` consistent with the `correct_option`.
```
````

B.2.7 FIXER

Fixer Prompt

````
You are an expert editor that executes a given fix plan with surgical
    precision. You will be given the original problem data and a set
    of instructions. Your task is to rewrite the specified fields to
    apply the fixes.

**Your Rules:**
- Only modify the fields explicitly mentioned in the instructions.
- If a field is not mentioned, do not change it.
- Apply ALL instructions in the plan.
- Do not add any new information, explanations, or stylistic changes.
    Your work should be a minimal-edit based on the plan.
- Do not remove, rename, or alter any image references. Preserve all
    image markdown and their order. Images are essential and must
    remain present in the corrected content.
- **CRITICAL JSON RULE:** The output must be a single, valid JSON
    object. The text fields (`problem`, `choices`, etc.) often contain
    markdown and LaTeX. In JSON strings, all backslash characters
    (`\\`) MUST be escaped with another backslash. For example, if the
    corrected text contains `\\binom{n}{k}`, you must write it as
    `\\\\binom{n}{k}` in the JSON output. This is the most important
    rule.

**Fix Plan:**
{fix_plan_md}

**Original Data:**
- problem: {problem}
- choices: {choices}
- english_solution_local_images: {english_solution}
- context: {context}
- correct_option: {correct_option}
- answer_value: {answer_value}
- SVG XMLs (if any):
  ```
  {svg_sources}
  ```

Note on SVGs: The SVG XML snippets are auxiliary. The equivalent PNG
    renderings are already present in the context. Use SVGs only to
    disambiguate equations or figure details while applying changes;
    do not output or modify SVGs.
````
````

```
Generate the 'FixedProblemData' as a single, valid JSON object that
    strictly conforms to the schema. Use double-quoted keys. For any
    fields you did not change, set them to null. Return only the JSON
    object  no schema, no prose, and no code fences.

**JSON Schema for Output:**
'''json
{
  "title": "FixedProblemData",
  "description": "The output from the Fixer stage, containing the
      complete, updated text for modified fields.",
  "type": "object",
  "properties": {
    "problem": {
      "type": ["string", "null"],
      "description": "The full, corrected problem text. If unchanged,
          this is null."
    },
    "choices": {
      "type": ["string", "null"],
      "description": "The full, corrected choices text. If unchanged,
          this is null."
    },
    "english_solution_local_images": {
      "type": ["string", "null"],
      "description": "The full, corrected solution text. If unchanged,
          this is null."
    },
    "context": {
      "type": ["string", "null"],
      "description": "The full, corrected context text. If unchanged,
          this is null."
    },
    "correct_option": {
      "type": ["integer", "null"],
      "description": "The corrected option number. If unchanged, this
          is null."
    },
    "answer_value": {
      "description": "The corrected answer value. If unchanged, this is
          null."
    }
  }
}
'''
```

B.2.8 Validator

Validator Prompt

```
You are a meticulous verifier and senior analyst. Your task is to
    validate that a set of fixes, applied to a math problem's data,
    has resolved the issues outlined in an original fix plan. If
    issues remain, you must create a new, refined fix plan.

### Governing Principles for Validation

Your analysis must be guided by the following strict principles. A
    fix is **invalid ('is_fixed: false')** if it violates any of them.

**1. Locational and Logical Integrity:**
```

```
*  A fix is **invalid** if the location of the change does not match
    the location of the reported error. You must first verify that the
    fields modified by the Fixer are the same fields where the error
    was identified in the `Original Issue Report`.
*  A fix is **invalid** if the *type* of fix is illogical for the
    *type* of error. For example, if the report identifies a
    `MistranslationEquation` in the solution, a fix that changes the
    `problem` text is logically inconsistent and must be rejected. The
    fix must directly address the reported issue in its specific
    context.

**2. Final Answer Integrity:**
Your verification of the final answer must follow two steps: checking
    permission and checking correctness.

*  **Permission Check:** First, check if `correct_option` or
    `answer_value` were modified. If they were, you must confirm that
    the original issue category was **`ManualErrorIncorrectGuess`**.
    Changing the final answer for any other reason is a critical
    failure and the fix is invalid.
*  **Correctness Check:**
   *  If the answer was changed (for a `ManualErrorIncorrectGuess`),
       you must verify that the new answer matches the ground truth
       derived from the `crawled_persian_markdown`'s proof (as a
       fallback for an ambiguous source).
   *  If the *proof text* was changed, you must verify that the new
       text now correctly derives the ground truth answer stated in
       the original Persian source's answer key. A fix is invalid if
       it "corrects" the proof to lead to the wrong answer.

**3. Scope of Edits (Minor Changes Only):**
*  You must ensure the Fixer did not perform a major rewrite of the
    solution. Compare the original and fixed
    `english_solution_local_images`. The changes should be minor and
    surgical (e.g., typos, variable corrections, a rewritten sentence
    or two). If the solution has been substantially rewritten, the fix
    is invalid.

**4. Content Preservation:**
*  You must verify that no important information, equations, or image
    references were accidentally deleted from the solution text. The
    fix should only add or modify, not remove correct information.

**Context:**
Another AI, the "Fixer," was given an original fix plan and the
    original problem data. It has produced a new version of the data.
    Your job is to act as a quality assurance step.

**CRITICAL: Understanding What the Fixer Can and Cannot Modify**
The Fixer can ONLY modify these specific fields:
- `problem` (the English problem statement)
- `choices` (the English choices)
- `english_solution_local_images` (the English solution)
- `context` (additional context text)
- `correct_option` (the correct option number)
- `answer_value` (the answer value)

The Fixer CANNOT and WILL NOT modify:
- `crawled_persian_markdown` (this is our source of truth and remains
    unchanged)
- Any other fields not listed above
```

```
When evaluating fixes, do NOT expect `crawled_persian_markdown` to be
    changed. It is provided only as a reference for comparison and
    validation purposes.

**Note on 'No Discernible Error' Category:** If the `Original Issue
    Report` states that the category is "No Discernible Error," it
    means the initial automated validation was likely a false
    positive. In this case, your primary task is to confirm that the
    problem data is indeed correct and that the "Fixer" has not
    introduced any unnecessary or incorrect changes. If the data
    remains correct, you should set `is_fixed` to `true`.

Note on Sources: The `crawled_persian_markdown` reflects what the
    official source published, but it may omit full solutions
    (sometimes only hints or only the problem). Treat it as
    authoritative for what it contains. When absent, a valid English
    solution may come from other trusted official materials (e.g.,
    official PDF extraction). Evaluate consistency across all provided
    materials and validation findings.

**Inputs:**

1. **Original Issue Report (`aggregated_report_md`):** This is the
    ground truth. It describes what was originally found to be wrong
    with the problem.
   {aggregated_report_md}

2. **Original Fix Plan (`fix_plan_md`):** The plan the Fixer was
    supposed to follow.
   {fix_plan_md}


3. **Original Problem Data:** The data before any changes were made.
   - **Problem:** {problem}
   - **Choices:** {choices}
   - **Solution:** {english_solution}
   - **Crawled Persian** Markdown (Source of Truth):
      {crawled_persian_markdown}
   - **Context:** {context}
   - **Correct Option:** {correct_option}
   - **Answer Value:** {answer_value}
   - **SVG XMLs (if any):**

     {svg_sources}


Note on SVGs: The SVG XML snippets are provided only to clarify
    equations or figure contents. The equivalent PNG images are
    already present in the data. Use SVGs as auxiliary references
    only; do not output or modify SVGs.

4. **Summary of Applied Fixes (`fixed_data_md`):** A summary of the
    changes the Fixer made.

   {fixed_data_md}


**Your Task:**

1. **Evaluate the Plan:** First, review the "Original Fix Plan." Does
    it seem like a reasonable and complete solution for the issues
    described in the "Original Issue Report"?
```

```
2. **Compare Data:** Meticulously compare the "Original Problem Data"
    with the "Summary of Applied Fixes." Remember: only evaluate
    changes to the fields the Fixer can modify (listed above). Do NOT
    expect `crawled_persian_markdown` to be changed.
3. **Verify:** Determine if the applied fixes successfully and
    completely address *all* the issues from the "Original Issue
    Report." Note any discrepancies between the plan and the final
    fix. Critically, ensure that all image references that existed in
    the original data are still present in the fixed content; if any
    image reference is missing, the fix must be rejected.
4. **Identify New Issues:** Check if the fixes introduced any new
    problems or cascading errors (e.g., changing the choices but not
    updating the `correct_option`).
5. **Make a Decision (`is_fixed`):**
   -  If all issues from the "Original Issue Report" are resolved and
       no new issues exist, set `is_fixed` to `true`.
   -  Otherwise, set `is_fixed` to `false`.
6. **Provide Reasoning:** Briefly explain your decision. If not
    fixed, clearly state what is still wrong, including any missing
    image references.
7. **Re-Plan Decision (`needs_replan`):**
   -  If `is_fixed` is `false` and the existing fix plan is
       inadequate or incorrect, set `needs_replan` to `true`.
   -  Otherwise, set `needs_replan` to `false`.

**Output Instructions:**
Produce a single, valid JSON object with double-quoted keys that
    conforms strictly to the schema below. Do NOT add any extra text,
    markdown, explanations, or code fences. Return only the JSON
    object.

Consistency Constraint (Critical):
- `is_fixed` can be `true` only and only if `needs_replan` is
    `false`. If `needs_replan` is `true`, then `is_fixed` must be
    `false`.

**CRITICAL JSON RULE:** The output must be a single, valid JSON
    object. Some fields may contain markdown and LaTeX. In any JSON
    string, all backslash characters (`\\`) MUST be escaped with
    another backslash. For example, if a fix plan instruction is
    `change \\frac to \\binom`, you must write it as "change \\\\frac
    to \\\\binom" in the JSON output. This is the most important rule.

**JSON Schema for Output:**

{
  "title": "ValidationResult",
  "type": "object",
  "properties": {
    "is_fixed": {
      "type": "boolean",
      "description": "True if all issues in the original plan are
          resolved and no new issues were created."
    },
    "reasoning": {
      "type": "string",
      "description": "A brief explanation of the validation outcome. If
          not fixed, this should explain what is still wrong."
    },
    "needs_replan": {
      "type": "boolean",
      "description": "True if the current fix plan should be revised
          before the next iteration."
```

```
    }
  },
  "required": ["is_fixed", "reasoning", "needs_replan"]
}
```

B.2.9 RePlanner

RePlanner Prompt

```
You are a meticulous technical editor and AI repair specialist. The
    Validator determined that the current fix plan needs revision.
    Write a new, clear, high-level, and machine-executable plan for
    the Fixer to carry out. The output must be a markdown document.

**How to Re-Assess the Issue: The Hierarchy of Suspicion**

The previous plan failed. Before creating a new one, you must
    re-evaluate the error's origin using the 'Aggregated Issue Report'
    and the 'Validator Reasoning'. Your new plan must be tailored to
    the error's source, following this hierarchy:

1. **If the error is from our Pipeline (Extraction/Translation):**
   *  **Your Goal:** Make our data a perfect reflection of the
       'crawled_persian_markdown' source.
   *  **Your Plan:** Create instructions to correct mistranslations,
       fix parsing errors, and align our data with the ground truth.

2. **If the error is a Minor Flaw in the Source Solution:**
   *  **Your Goal:** Correct the minor flaw (e.g., typo, notational
       error) in the source's logic and reflect that fix in our
       English data.
   *  **Your Plan:** Your instructions should surgically correct the
       'english_solution_local_images' to fix the issue.

3. **If there are Combined Errors (Source + Pipeline):**
   *  **Your Goal:** Create a plan that addresses the root cause
       first.
   *  **Your Plan:** Your instructions must be ordered correctly.
       First, an instruction to address the conceptual fix needed for
       the source error. Second, an instruction to fix the translation
       based on that now-corrected concept.

### CRITICAL RULES FOR PLANNING FIXES

Your authority to make changes is strictly limited. While your
    primary goal is to create a complete plan to fix all issues in the
    report, you must operate within the following non-negotiable
    constraints:

#### RULE 0: CONFLICT RESOLUTION
Your primary goal is to follow all rules. If you find that fixing an
    issue according to one rule (e.g., 'RULE 3') would force you to
    violate another rule (e.g., 'RULE 1'), you must prioritize safety.
    Your plan should:
1. Perform any minor, safe fixes that do not cause a conflict.
2. Clearly state the nature of the rule conflict you encountered
    (e.g., "Correcting the solution to match the updated problem would
    require a full rewrite, which violates RULE 1.").
3. Explicitly recommend that the problem requires human intervention.

#### RULE 1: MODIFICATIONS MUST BE MINOR AND SURGICAL
```

```
You are **forbidden** from rewriting entire solutions. The goal is to
    repair, not replace.
*  **You CAN:** Make minor edits like correcting typos, changing
    variables, fixing indices, or modifying equations within a
    sentence. You may rewrite one or two sentences if absolutely
    necessary to correct a specific, localized error.
*  **You CANNOT:** Propose a total rewrite, restructure the entire
    logical flow, or add large new paragraphs of explanation.

#### RULE 2: THE FINAL ANSWER IS SACROSANCT

You are **strictly forbidden** from planning any changes to
    `correct_option` or `answer_value` unless the aggregated issue
    report's final category is exactly **`ManualErrorIncorrectGuess`**.

*  **IF the category is `ManualErrorIncorrectGuess`:** Your plan's
    objective is to derive the correct answer from the mathematical
    proof in the `crawled_persian_markdown` and update
    `correct_option` and `answer_value` to match that derived truth.
*  **IF the category is `OriginalSourceError`:** You **must not**
    change `correct_option` or `answer_value`. Your plan must focus on
    making minor textual edits to the solution to clarify the flawed
    reasoning or fix the notation/typos.
*  **IF the category is `MistranslationAnswerKey`:** Your plan must
    **only** remove the sentence stating the answer is not in the
    choices. Do not change `correct_option` or `answer_value`.

#### RULE 3: UPHOLD THE HIERARCHY OF TRUTH

Your primary directive is to ensure the data is a high-fidelity
    representation of the original Persian source
    (`crawled_persian_markdown`). All fixes must follow this strict
    hierarchy, where lower-priority data is always corrected to match
    higher-priority data.

1. **Ultimate Authority (`crawled_persian_markdown`):** This is the
    absolute ground truth.
2. **Problem Definition (`problem`, `context`, `choices`):** These
    fields must be a faithful translation of the Ultimate Authority.
3. **Derived Explanation (`english_solution_local_images`):** This
    field must correctly solve the problem as defined in the `problem`
    field.

- **You MUST:** If the `context` contains a typo or mistranslation
    (when compared to the Ultimate Authority), your plan must correct
    the `context` field.
- **You MUST:** If the `problem` has a typo or mistranslation (when
    compared to the Ultimate Authority), your plan must correct the
    `problem` field AND then also correct the
    `english_solution_local_images` so it solves the now-correct
    problem.
- **You MUST NOT:** Ever "fix" the `problem` field to justify an
    error in the `english_solution_local_images`. The solution always
    yields to the problem.

Any plan that violates these rules is invalid and will be rejected.

Inputs:
- Aggregated Issue Report (markdown):
  {aggregated_report_md}
- Validator Reasoning (why previous plan failed):
  {validator_reasoning}
```

```
- Existing Fix Plan (to revise):
  {fix_plan_md}

Text Fields to Analyze:
- problem: {problem}
- choices: {choices}
- english_solution_local_images: {english_solution}
- context: {context}
- correct_option: {correct_option}
- answer_value: {answer_value}
- SVG XMLs (if any):
  ```
  {svg_sources}
  ```

Constraints (Critical):
- Do not propose removing, renaming, or altering any image
    references. Image content is essential and must be preserved.
- If an instruction would implicitly remove an image, rewrite it to
    keep images intact and only change necessary text.
- Never instruct to delete image markdown (e.g., lines that start
    with `![](` or similar).

Required Output Structure:
```markdown
### Fix Plan

1. **Instruction:** [...]
   *  **Target Fields:** [...]
   *  **Rationale:** [...]
2. **Instruction:** [...]
   *  **Target Fields:** [...]
   *  **Rationale:** [...]
```
```

C Complete Technique Taxonomy

The following hierarchy contains all 89 sub-sub-topic labels used for technique classification in CombiGraph-Vis. Each problem receives labels from this taxonomy based on techniques that explicitly appear in its solution.

C.1 Technique Labeling Prompt

Technique Labeler Prompt

```
# Task

Given a `{problem}`, its `{solution}`, and optional `{context}`,
    determine which techniques were **actually used** in the solution
    and output them as a **list** of labels. Each label must strictly
    follow the three-level path:

`Topic -> Sub-topic -> Sub-sub-topic`

Only use items from the **Reference Topic Hierarchy** below. Pick the
    **most specific** sub-sub-topic(s) that apply.

# Inputs
```

````
* **Problem:** `{problem}`
* **Solution:** `{solution}`
* **Context (optional):** `{context}`

## What Context Means (read carefully)

* **Definition:** `{context}` is any preliminary text that defines
    the setting, objects, constraints, notations, or assumptions that
    the problem and solution rely on (e.g., colors are considered
    identical up to rotation, multisets allowed, graph is simple and
    undirected, special definitions, or domain restrictions).
* **Usage Rule:** Treat `{context}` as part of the problem setup. If
    `{context}` narrows, extends, or clarifies the setting, **apply it
    when deciding techniques** (e.g., combinations with repetition
    becomes applicable if `{context}` allows multisets).
* **Conflict Rule:** If `{context}` conflicts with generic
    assumptions, **prefer `{context}`** unless the solution explicitly
    overrides it.

# Decision Rules (strict)

1. **Most-specific only:** Every label must be a full three-level
    chain from the hierarchy (no truncations).
2. **Evidence-based:** Base labels on steps that *appear in the
    solution*, not merely plausible alternatives.
3. **Context-aware:** Incorporate `{context}` constraints/definitions
    when identifying techniques.
4. **Multi-technique:** Include all materially used techniques. Mark
    exactly one label as primary.
5. **Ties:** If two sub-sub-topics plausibly apply, prefer the one
    explicitly named or most central to the argument.
6. **Out-of-scope moves:** If the solution uses ideas not present in
    the hierarchy, add one extra array item with `"topic": "OTHER"`
    and a short `"justification"` describing the idea. Do **not**
    invent new hierarchy items.

# Output Format (JSON)

Return **only** a JSON **array**. Each element is an object of this
    shape:

```json
[
  {
    "topic": "",
    "sub_topic": "",
    "sub_sub_topic": "",
    "primary": true,
    "justification": "13 sentences citing the exact step(s) in the
        solution (and any relevant context) that evidence this
        technique."
  }
]
```

* Include **exactly one** element with `"primary": true`. All others
    must have `"primary": false`.
* If there are no valid hierarchy techniques, return an array with a
    single `"OTHER"` item as described in Rule 6.

# Worked Micro-Examples

**Example A (single technique)**
````

````
Solution step: We count integer solutions to $x_1+\dots+x_k=n$ using
    stars and bars.
    Output:

```json
[
  {
    "topic": "Combinatorics",
    "sub_topic": "Counting Foundations",
    "sub_sub_topic": "Stars & bars",
    "primary": true,
    "justification": "Applies the balls-into-bins formula to count
        nonnegative integer solutions to a sum."
  }
]
```

**Example B (multiple techniques)**
Solution steps: Apply InclusionExclusion to avoid overcounting then
    use linearity of expectation to bound the count.
    Output:

```json
[
  {
    "topic": "Combinatorics",
    "sub_topic": "Advanced Counting",
    "sub_sub_topic": "InclusionExclusion (e.g., derangements)",
    "primary": true,
    "justification": "Main count constructed via inclusionexclusion to
        correct overcounting."
  },
  {
    "topic": "Combinatorics",
    "sub_topic": "Probabilistic Method (intro)",
    "sub_sub_topic": "Linearity-of-expectation tricks",
    "primary": false,
    "justification": "Uses expectation linearity to bound the count
        after inclusionexclusion."
  }
]
```

# Reference Topic Hierarchy (choose **only** from these leaves)

## Combinatorics

* **Counting Foundations**

  * Sum/Product/Complement rules
  * Bijections (one-to-one counting)
  * Permutations & arrangements (with/without repetition; circular)
  * Combinations (with/without repetition; multisets)
  * Stars & bars (integer-solution counting)
  * Binomial theorem; lattice paths; basic identities
* **Advanced Counting**

  * InclusionExclusion (e.g., derangements)
  * Double counting
  * **Recurrences & Generating Ideas**

    * Linear recurrences (characteristic equations)
    * Classic sequences (Fibonacci, Catalan)
````

```
    * Light generating functions (ordinary/exponential)
  * **Symmetry Counting**

    * Burnsides lemma
    * Plya enumeration (intro)
* **Invariants & Monovariants**

  * Parity/modular invariants
  * Coloring/weighting arguments
  * Termination via monovariants
* **Probabilistic Method (intro)**

  * Linearity-of-expectation tricks
  * Existence proofs via expectation

## Graph Theory

* **Basics**

  * Definitions & representations (adjacency list/matrix)
  * Degree/handshaking; degree & *graphic* sequences
  * Isomorphism; traversals (BFS/DFS); paths, cycles, distance
* **Trees**

  * Properties; rooted/binary trees
  * DFS/BFS trees
  * Spanning trees & counting
* **Connectivity**

  * Connectedness; cut vertices/bridges
  * k-connectivity; blocks (biconnected components)
* **Directed Graphs**

  * Strongly connected components
  * Tournaments
* **Cycles & Trails**

  * Eulerian trails/tours
  * Hamiltonian paths/cycles
* **Matchings & Covers**

  * Bipartite matchings; Halls marriage theorem
  * Matchings in general graphs; independence number
  * Vertex/edge covers (and relations in bipartite graphs)
* **Planarity & Coloring**

  * Planar graphs; Eulers formula (applications)
  * Vertex/edge coloring; counting colorings

## Combinatorial Game Theory

* **Modeling & State Analysis**

  * Game graphs; win/lose/draw states
  * DP for state evaluation; kernels; strategy existence proofs
* **Canonical Examples**

  * Nim; partisan games; Hex; Shannon switching game

## Probability (Elementary)

* **Core Concepts**
```

```
  * Sample spaces & events; basic probability
  * Conditional probability; independence; Bernoulli trials
* **Expectation**

  * Random variables; linearity of expectation
  * Indicator variables

## Number Theory (Contest Essentials)

* **Divisibility & GCD/LCM**

  * Euclidean algorithm; Bzouts identity
* **Primes & Congruences**

  * Modular arithmetic; Fermats little theorem; CRT
* **Counting Toolbox**

  * Multiplicative functions (n), (n), (n); multiplicativity
  * Fast exponentiation; modular inverses
  * Counting by gcd/lcm; CRT-based counts

## Formal Languages & Automata (CS touch-in)

* **Languages**

  * Alphabets, strings, languages
* **Machines**

  * DFA & NFA; pushdown automata; Turing machines

## Algorithmic Techniques (non-coding)

* **Greedy**

  * Exchange arguments; counterexample design
* **Dynamic Programming**

  * State modeling for counting/optimization (sequences, grids, graphs)
* **Divide-and-Conquer & Recursion**

  * Recurrences; correctness ideas
* **Search**

  * Backtracking & pruning; BFS/DFS as search patterns
* **Classic Tricks**

  * Binary search on answer; two-pointers/sliding window
* **Proof of Correctness**

  * Invariants; loop/phase arguments

## Conceptual Data Structures (no code)

* **Linear Containers**

  * Stack, queue, deque
* **Priority & Set Structures**

  * Heaps/priority queues; sets/maps; hashing ideas
* **Disjoint Set Union (UnionFind)**

  * Connectivity; cycle detection
* **Graph Representations**
```

```
  * Adjacency list vs matrix; trade-offs

## Strings & Combinatorics on Words

* **Structural Properties**

  * Prefix/suffix/border; periodicity
  * Palindromes
* **Counting & Constraints**

  * Counting constrained strings
  * Links to automata (acceptance as constraints)

## Discrete and Computational Geometry

* **Primitives**

  * Orientation test (cross-product sign)
  * Line/segment intersection
* **Polygons & Lattice**

  * Polygon area (shoelace)
  * Lattice points; Picks theorem
* **Convexity**

  * Convex-hull intuition and uses

## Logical & Puzzle Reasoning

* **Logic & Proof Moves**

  * Propositional logic; contradiction/contrapositive
* **Puzzle Tactics**

  * Invariants for grid/tiling; parity tricks
  * Constructive examples & counterexamples

## Inequalities & Algebraic Tools

* **Core Inequalities**

  * AMGM; CauchySchwarz (incl. Titus lemma)
  * Rearrangement inequality
* **Summation Tricks**

  * Telescoping; bounding techniques

## General Proof Strategies

* **Mathematical Induction**

  * Weak vs. Strong induction
  * Structural induction (on trees, graphs, etc.)
  * Formulating & strengthening the inductive hypothesis
  * Infinite descent / Minimal counterexample
* **Pigeonhole Principle (PHP)**

  * Simple form (n+1 pigeons in n holes)
  * Generalized/Strong form (\$\lceil N/k \rceil\$ items)
  * Applications in geometry, number theory, and graphs
* **Extremal Principle**
```

```
  * Core idea (Max/Min argument)
  * Proving existence or properties of extremal objects
* **Coloring & Invariant Arguments**

  * Coloring proofs (e.g., checkerboard/parity coloring)
  * Invariants (properties that remain constant)
  * Monovariants (properties that change monotonically)

---
```

C.2 Solution Generation Prompt

Solution Generation Prompt

```
# Olympiad Problem Solution Instructions

You are tasked with solving a mathematical olympiad-level problem.
    Provide a complete, rigorous, and mathematically accurate solution
    that meets the standards expected in competitive mathematics.

## Input Components

**Context:** {context}
- This provides background information, definitions, and preliminary
    setup for the problem
- Pay careful attention to any special notation, constraints, or
    conditions defined here

**Problem:** {problem}
- This is the main question to be solved
- Identify exactly what is being asked and what the final answer
    should be

**Choices:** {choices}
- If present, these are the multiple choice options
- Your final answer must match one of these choices exactly


## Solution Standards

Your solution must demonstrate:

1. **Complete Mathematical Rigor**: Every step must be mathematically
    justified with proper reasoning
2. **Clear Logical Flow**: Present arguments in a logical sequence
    that builds toward the solution
3. **Precise Definitions**: Use mathematical terminology accurately
    and define any non-standard notation
4. **Thorough Analysis**: Consider all relevant cases and address
    potential edge cases
5. **Computational Accuracy**: All calculations must be correct and
    verifiable
6. **Proof Completeness**: If proving a statement, ensure the proof
    covers all necessary cases and is gap-free

## Solution Structure

1. **Problem Analysis**: Begin by clearly restating what needs to be
    found and identifying key constraints
2. **Approach Strategy**: Explain your solution method and why it's
    appropriate
```

```
3. **Detailed Working**: Show all mathematical steps with clear
   justifications
4. **Verification**: When possible, verify your answer through
   alternative methods or checking edge cases
5. **Final Answer**: Present the final answer clearly

## Mathematical Notation Requirements

- Use correct LaTeX notation for all equations and mathematical
   symbols
- Use `\\(` and `\\)` for inline mathematics
- Use `\\[` and `\\]` for display mathematics (block equations)
- Do not use any unicode characters - stick to proper LaTeX formatting
- Show intermediate steps clearly with proper mathematical formatting

## Answer Format Requirements

- Wrap your final numerical answer, expression, or choice in:
   `\boxed{your_answer}`
- For multiple choice questions, include both the choice number and
   description if applicable
- Ensure the boxed answer directly addresses what the problem asks for
- If the answer is a mathematical expression, present it in its
   simplest form

## Mathematical Communication

- Use proper mathematical terminology and maintain precision in
   language
- Distinguish clearly between "implies," "if and only if," "for all,"
   etc.
- Explain the reasoning behind each major step
- Present arguments in a logical sequence that builds toward the
   solution
- Consider all relevant cases and address potential edge cases

Solve the given problem following these guidelines.
```

C.3 Hierarchical Taxonomy

C.4 Combinatorics

Counting Foundations

- Sum/Product/Complement rules
- Bijections (one-to-one counting)
- Permutations & arrangements (with/without repetition; circular)
- Combinations (with/without repetition; multisets)
- Stars & bars (integer-solution counting)
- Binomial theorem; lattice paths; basic identities

Advanced Counting

- InclusionExclusion (e.g., derangements)
- Double counting

Recurrences & Generating Ideas

- Linear recurrences (characteristic equations)

- Classic sequences (Fibonacci, Catalan)
- Light generating functions (ordinary/exponential)

Symmetry Counting

- Burnside's lemma
- Plya enumeration (intro)

Invariants & Monovariants

- Parity/modular invariants
- Coloring/weighting arguments
- Termination via monovariants

Probabilistic Method (intro)

- Linearity-of-expectation tricks
- Existence proofs via expectation

C.5 Graph Theory

Basics

- Definitions & representations (adjacency list/matrix)
- Degree/handshaking; degree & graphic sequences
- Isomorphism; traversals (BFS/DFS); paths, cycles, distance

Trees

- Properties; rooted/binary trees
- DFS/BFS trees
- Spanning trees & counting

Connectivity

- Connectedness; cut vertices/bridges
- k-connectivity; blocks (biconnected components)

Directed Graphs

- Strongly connected components
- Tournaments

Cycles & Trails

- Eulerian trails/tours
- Hamiltonian paths/cycles

Matchings & Covers

- Bipartite matchings; Hall's marriage theorem
- Matchings in general graphs; independence number
- Vertex/edge covers (and relations in bipartite graphs)

Planarity & Coloring

- Planar graphs; Euler's formula (applications)
- Vertex/edge coloring; counting colorings

C.6 COMBINATORIAL GAME THEORY

Modeling & State Analysis

- Game graphs; win/lose/draw states
- DP for state evaluation; kernels; strategy existence proofs

Canonical Examples

- Nim; partisan games; Hex; Shannon switching game

C.7 PROBABILITY (ELEMENTARY)

Core Concepts

- Sample spaces & events; basic probability
- Conditional probability; independence; Bernoulli trials

Expectation

- Random variables; linearity of expectation
- Indicator variables

C.8 NUMBER THEORY (CONTEST ESSENTIALS)

Divisibility & GCD/LCM

- Euclidean algorithm; Bzout's identity

Primes & Congruences

- Modular arithmetic; Fermat's little theorem; CRT

Counting Toolbox

- Multiplicative functions (n), (n), (n); multiplicativity
- Fast exponentiation; modular inverses
- Counting by gcd/lcm; CRT-based counts

C.9 FORMAL LANGUAGES & AUTOMATA (CS TOUCH-IN)

Languages

- Alphabets, strings, languages

Machines

- DFA & NFA; pushdown automata; Turing machines

C.10 ALGORITHMIC TECHNIQUES (NON-CODING)

Greedy

- Exchange arguments; counterexample design

Dynamic Programming

- State modeling for counting/optimization (sequences, grids, graphs)

Divide-and-Conquer & Recursion

- Recurrences; correctness ideas

Search

- Backtracking & pruning; BFS/DFS as search patterns

Classic Tricks

- Binary search on answer; two-pointers/sliding window

Proof of Correctness

- Invariants; loop/phase arguments

C.11 CONCEPTUAL DATA STRUCTURES (NO CODE)

Linear Containers

- Stack, queue, deque

Priority & Set Structures

- Heaps/priority queues; sets/maps; hashing ideas

Disjoint Set Union (UnionFind)

- Connectivity; cycle detection

Graph Representations

- Adjacency list vs matrix; trade-offs

C.12 STRINGS & COMBINATORICS ON WORDS

Structural Properties

- Prefix/suffix/border; periodicity
- Palindromes

Counting & Constraints

- Counting constrained strings
- Links to automata (acceptance as constraints)

C.13 DISCRETE AND COMPUTATIONAL GEOMETRY

Primitives

- Orientation test (cross-product sign)
- Line/segment intersection

Polygons & Lattice

- Polygon area (shoelace)
- Lattice points; Pick's theorem

Convexity

- Convex-hull intuition and uses

C.14 Logical & Puzzle Reasoning

Logic & Proof Moves

- Propositional logic; contradiction/contrapositive

Puzzle Tactics

- Invariants for grid/tiling; parity tricks
- Constructive examples & counterexamples

C.15 Inequalities & Algebraic Tools

Core Inequalities

- AMGM; CauchySchwarz (incl. Titu's lemma)
- Rearrangement inequality

Summation Tricks

- Telescoping; bounding techniques

C.16 General Proof Strategies

Mathematical Induction

- Weak vs. Strong induction
- Structural induction (on trees, graphs, etc.)
- Formulating & strengthening the inductive hypothesis
- Infinite descent / Minimal counterexample

Pigeonhole Principle (PHP)

- Simple form (n+1 pigeons in n holes)
- Generalized/Strong form ($\lceil N/k \rceil$ items)
- Applications in geometry, number theory, and graphs

Extremal Principle

- Core idea (Max/Min argument)
- Proving existence or properties of extremal objects

Coloring & Invariant Arguments

- Coloring proofs (e.g., checkerboard/parity coloring)
- Invariants (properties that remain constant)
- Monovariants (properties that change monotonically)