Supplementary Material for FinMMR

Contents

A Error Analysis

We exhibit representative examples for the three primary error categories: **Visual Perception Error** (Fig. 2), **Knowledge Reasoning Error** with two cases (Fig. 3, Fig. 4), and **Numerical Computation Error** (Fig. 5). These errors were observed in the outputs of Claude 3.7 Sonnet with 64K extended thinking in PoT setting.

These error cases highlight critical limitations in current multimodal large language models (MLLMs). **Visual Perception** errors primarily stem from the misinterpretation of complex visual data (such as charts) or failures to accurately extract numerical information from tables. **Knowledge Reasoning** errors result from either incorrect factual knowledge stored in the model or flawed logical reasoning processes. **Numerical Computation** errors occur due to inaccuracies in the model's internal computation mechanisms during reasoning. Consequently, FinMMR underscores the need for improvements in three core capabilities: intricate visual perception, specialized knowledge reasoning, and accurate numerical computation.

B Augmentation Methods

B.1 Visual Filtering for Reasoning

Fig. 6 illustrates a representative example demonstrating our proposed two-stage filtering-reasoning pipeline applied to a multimodal numerical reasoning task. Specifically, the task requires calculating fifth-year sales revenue for the respiratory category, comparing it with third-year data, and computing the growth rate. Among the three input images, Images 1 and 2 contain task-relevant tables, while Image 3 is an irrelevant distractor. Without visual filtering, although the model correctly avoided extracting numerical information from the irrelevant image, it misinterpreted values from relevant tables, resulting in a significant deviation from the reference answer. When implementing our pipeline, the MLLM first assesses image relevance, correctly identifying Images 1 and 2 as `[USEFUL]` and Image 3 as `[USELESS]`, thus eliminating distractors from downstream reasoning.

B.2 Knowledge Augmentation

Fig. 7 presents a representative financial multimodal reasoning task, demonstrating the efficacy of our domain knowledge augmentation method. Specifically, the task requires calculating the impact on net income from switching inventory accounting methods from LIFO to FIFO, based on a provided financial table. In the baseline without augmentation ("Original Output"), the model incorrectly treated the entire 2014 LIFO reserve as income, violating U.S. GAAP which stipulates that **only changes in LIFO reserve** impact current-year net income. This caused significant income overstatement. Conversely, with our augmentation approach ("Augmented Output"), the model correctly computed the effect using the **year-over-year change in LIFO reserve** with proper tax effect adjustment, achieving accurate income quantification. This case highlights the critical importance of domain knowledge augmentation for MLLMs. Complementing this, Fig. 8 illustrates a representative financial function retrieved from our library.

B.3 Visual Parser with Reasoner

Fig. 9 demonstrates our two-stage parsing-reasoning pipeline comprising a **Visual Parser** and a **Reasoner**. The task requires calculating anticipated portfolio returns under two economic scenarios using probability-weighted returns from a tabular image. We first instruct GPT-4o (as Visual Parser) to convert tabular data into structured markdown format. This structured output then enables a large reasoning model (LRM) to perform numerical reasoning. Compared to directly reasoning on raw visual inputs, GPT-4o's parsed data significantly enhances the downstream LRM's accuracy, underscoring the efficacy of model collaboration in multimodal reasoning.

C Prompt Templates

C.1 Chain-of-Thought (CoT)

The CoT prompt directs MLLMs to decompose complex reasoning tasks into sequential intermediate steps. This approach enhances performance on mathematical, logical, and analytical problems by requiring explicit articulation of each reasoning stage before final answer generation. See Fig. 10 for the complete template.

C.2 Program-of-Thought (PoT)

The PoT prompt guides MLLMs to formulate solutions as executable code. By translating problems into programs with variables, functions, and logical operations, this approach enables precise computation and transparent reasoning, particularly effective for quantitative tasks. The complete template is provided in Fig. 11.

C.3 Retrieval-Augmented Generation (RAG)

The RAG prompt integrates retrieval of external knowledge with generative reasoning. This hybrid approach enables MLLMs to supplement parametric knowledge with dynamically retrieved information, enhancing answer accuracy for knowledge-intensive tasks. See Fig. 12 for implementation details.

C.4 Function Relevance Judgment

This prompt evaluates the utility of retrieved financial functions for answering specific questions. It assesses both relevance and reliability, permitting only qualified knowledge to inform downstream reasoning. The complete template is available in Fig. 13.

C.5 Visual Relevance Judgment

This prompt assesses image relevance for financial problem-solving. It classifies each input image as `[USEFUL]` or `[USELESS]`, with uncertainty favoring retention to prevent critical information loss. See Fig. 14 for details.

D Experimental Details

D.1 Model Specifications

Tab. 1 details the 15 MLLMs evaluated on FinMMR, covering models from 8 leading organizations. Each entry includes the official model identifier to ensure full experimental reproducibility.

D.2 Validation Set Results

Tab. 2 presents the accuracy metrics of 12 MLLMs on the **validation** set of FinMMRacross three difficulty tiers. Models are evaluated using both Chain-of-Thought (CoT) and Program-of-Thought (PoT) prompting methods under standardized conditions.

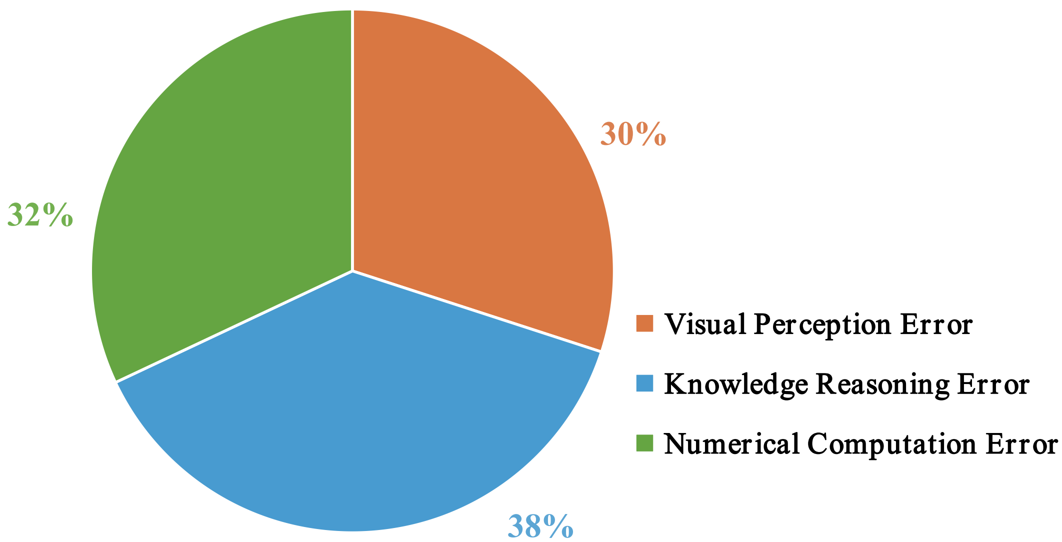

Figure 1: Error Distribution: Claude 3.7 Sonnet on FinMMR

Question: What is the increase in the percentage of the total contract assets proportion to the total accounts receivable from December 31, 2019 to June 30, 2020 in percentage? Rounded to 2 decimal places.

Accounts Receivable

The components of accounts receivable are as follows:

	As of	
	June 30, 2020	December 31, 2019
Trade receivables	$ 4,592	$ 6,425
Unbilled receivables	1,763	1,857
Other short-term receivables	19	7
Allowance for doubtful accounts	(716)	(1,816)
Total accounts receivable	$ 5,658	$ 6,473

Context: The timing of revenue recognition, billings and cash collections results in billed accounts receivable, unbilled receivables (contract assets), and customer advances and deposits (contract liabilities) on the consolidated balance sheets. In our Air Pollution Control Technology segment, amounts are billed as work progresses in accordance with agreed-upon contractual terms. Generally, billing occurs subsequent to revenue recognition, resulting in contract assets. These assets are reported on the consolidated balance sheet on a contract-by-contract basis at the end of each reporting period. At June 30, 2020 and December 31, 2019, contract assets were approximately $1,763 and $1,857, respectively, and are included in accounts receivable on the consolidated balance sheets.

```
def solution():
    # Define variables name and value
    contract_assets_june_2020 = 1763
    total_accounts_receivable_june_2020 = 5058 (×)  5058 (×) —— 5658 (√)
  contract_assets_dec_2019 = 1857
    total_accounts_receivable_dec_2019 = 6473

    # Calculate the proportion of contract assets to total accounts
receivable
    proportion_june_2020 = contract_assets_june_2020 /
total_accounts_receivable_june_2020
    proportion_dec_2019 = contract_assets_dec_2019 /
total_accounts_receivable_dec_2019

    # Compute the percentage increase in proportion
    percentage_increase = ((proportion_june_2020 - proportion_dec_2019)
/ proportion_dec_2019) * 100

    # Return answer rounded to 2 decimal places
    return round(percentage_increase, 2)
```

Answer: 21.5

Ground Truth: 2.47

Figure 2: Visual Perception Error Case

Question: For each of the following $1,000-parvalue bonds, assuming annual interest payment and a 40% tax rate, calculate the after-tax cost to maturity using the approximation formula. What is the after-tax cost to maturity for Bond C? Answer as a percentage to two decimal places.

Bond	Life (years)	Underwriting fee	Discount (−) or premium (+)	Coupon interest rate
A	20	$25	−$20	9%
B	16	40	+10	10
C	15	30	−15	12
D	25	15	Par	9
E	22	20	−60	11

```
def solution():
    # Define bond parameters
    par_value = 1000  # Par value of the bond
    discount = -15  # Discount for Bond C ($15)
    life = 15  # Bond life in years
    coupon_rate = 0.12  # Coupon interest rate (12%)
    tax_rate = 0.40  # Tax rate (40%)
    underwriting_fee = 30  # Underwriting fee ($30)
    annual_interest = par_value * coupon_rate
    after_tax_interest = annual_interest * (1 - tax_rate) (×)
    net_proceeds = par_value + discount - underwriting_fee
    numerator = after_tax_interest + ((par_value - net_proceeds) / life)
 (×)
    denominator = (par_value + net_proceeds) / 2
    after_tax_cost = (numerator / denominator) * 100  # Convert to
percentage
    return round(after_tax_cost, 2)
```

Error Analysis: The error occurred because the tax rate was applied prematurely to the interest payment, while the amortized issuance cost was left untaxed, leading to an incorrect calculation. The correct approach is to first compute the pre-tax cost and then apply the tax rate to the entire cost to ensure consistency.

Ground Truth: 7.55

Figure 3: Knowledge Reasoning Error Case 1

Question: What percentage of the total leased locations are in Texas? Answer to three decimal places.

State	Number of Locations(1)
California	57
Florida	26
Georgia	22
Texas	19
Minnesota, New York	9
Illinois, Ohio, Maryland	8
Pennsylvania	7
Other	63

```
def solution():
    # Define the number of locations in different states
    locations = {
        "California": 57,
        "Florida": 26,
        "Georgia": 22,
        "Texas": 19,
        "Minnesota, New York": 9, (×)
        "Illinois, Ohio, Maryland": 8, (×)
     "Pennsylvania": 7,
        "Other": 63
    }
    # Calculate the total number of locations
    total_locations = sum(locations.values())
    # Compute the percentage of locations in Texas
    texas_percentage = (locations["Texas"] / total_locations) * 100
    # Return the result rounded to 3 decimal places
    return round(texas_percentage, 3)
```

Error Analysis: The error reflects a model reasoning issue, where multiple distinct states are incorrectly grouped as single entities—"Minnesota, New York" as one instead of two, and "Illinois, Ohio, Maryland" as one instead of three. This misclassification can lead to flawed inferences and inaccuracies in downstream analysis. In addition to the data shown in the image, the context also specifies that the company leases approximately 81 locations outside the United States. Taken together with the U.S.-based leased properties listed in the image, this brings the total number of leased locations to 292.

Ground Truth: 6.507

Figure 4: Knowledge Reasoning Error Case 2

Question: On February 17, 2017, what was the company's market capitalization as reported on the NYSE? Answer to three decimal places.

2016	High	Low
Quarter ended March 31	$102.93	$83.07
Quarter ended June 30	113.63	101.87
Quarter ended September 30	118.26	107.57
Quarter ended December 31	118.09	99.72
2015	High	Low
Quarter ended March 31	$101.88	$93.21
Quarter ended June 30	98.64	91.99
Quarter ended September 30	101.54	86.83
Quarter ended December 31	104.12	87.23

Context: Part II, Item 5. Market for Registrant's Common Equity, Related Stockholder Matters and Issuer Purchases of Equity Securities. The following table presents the reported quarterly high and low per share sale prices of our common stock on the NYSE for the years 2016 and 2015. On February 17, 2017, the closing price of our common stock was $108.11 per share as reported on the NYSE. As of February 17, 2017, we had 427,195,037 outstanding shares of common stock and 153 registered holders.

```
def calculate_market_cap():
  closing_price_per_share = 108.11
    outstanding_shares = 427_195_037
    market_capitalization = closing_price_per_share *
outstanding_shares
    market_cap_billions = round(market_capitalization /
1_000_000_000, 3) (×)
    return market_cap_billions
print(calculate_market_cap())
```

Error Analysis: The model arbitrarily converted the market capitalization into billions, despite the absence of any such requirement in the problem statement. This reflects a misalignment between the model's assumptions and the explicit task constraints.

Ground Truth: 46184055450.078

Figure 5: Numerical Computation Error Case

Question: 请根据图中数据，计算呼吸在第五年的销售额，并将其与第三年进行比较，计算增长率；请保留两位小数。
(Calculate the sales revenue of the “Respiratory” category in the fifth year based on the data in the table. Then, compare it with the third year and calculate the growth rate. Please round the result to two decimal places.)

Image 1:

	2021	2022	2023	2024E	2025E	2026E
总收入	1118.57	1363.35	1387.74	1542.15	1818.67	2117.58
YOY	-1.07%	21.88%	1.79%	11.13%	17.93%	16.44%
毛利率	42.36%	43.19%	45.89%	43.76%	43.94%	44.42%
归母净利润	106.09	166.57	192.49	220.85	269.86	323.18
YOY	-30.71%	57.00%	15.56%	14.74%	22.19%	19.76%
1.麻醉	349.08	409.45	467.73	493.46	592.15	710.58
YOY	8.67%	17.30%	14.23%	5.50%	20.00%	20.00%
毛利率	53.27%	54.81%	56.41%	56.00%	56.00%	56.00%
占比	31.21%	30.03%	33.70%	32.00%	32.56%	33.56%
2.导尿	290.40	424.19	382.21	443.36	523.17	627.80
YOY	5.24%	46.07%	-9.90%	16.00%	18.00%	20.00%
毛利率	27.16%	28.05%	29.27%	30.00%	30.00%	30.00%

USEFUL

Image 2:

占比	25.96%	31.11%	27.54%	28.75%	28.77%	29.65%
3.泌尿外科	169.56	196.04	203.44	189.20	223.26	263.44
YOY	27.84%	15.62%	3.77%	-7.00%	18.00%	18.00%
毛利率	80.35%	80.15%	75.90%	76.00%	76.00%	76.00%
占比	15.16%	14.38%	14.66%	12.27%	12.28%	12.44%
4.护理	134.02	159.44	135.13	189.18	227.01	215.66
YOY	39.52%	18.96%	-15.25%	40.00%	20.00%	-5.00%
毛利率	19.17%	23.20%	28.24%	25.00%	25.00%	25.00%
占比	11.98%	11.69%	9.74%	12.27%	12.48%	10.18%
5.呼吸	61.87	64.33	87.73	100.01	120.01	150.01
YOY	-9.87%	3.98%	36.36%	14.00%	20.00%	25.00%
毛利率	31.58%	37.69%	38.31%	37.00%	37.00%	37.00%
占比	5.53%	4.72%	6.32%	6.48%	6.60%	7.08%
6.血液透析	73.11	81.02	68.46	75.30	82.83	91.12
YOY	3.74%	10.82%	-15.51%	10.00%	10.00%	10.00%
毛利率	26.41%	24.46%	29.51%	28.00%	28.00%	28.00%
占比	6.54%	5.94%	4.93%	4.88%	4.55%	4.30%

USEFUL

Image 3:

公司名称	代码	收盘价（元）	归母净利润（亿元）				PE		
			23A	24E	25E	26E	24E	25E	26E
康德莱	603987.SH	7.46	2.31	2.24	2.48	2.81	14.23	12.84	11.32
三鑫医疗*	300453.SZ	7.74	2.07	2.45	2.86	3.36	16.69	14.30	12.17
拱东医疗	605369.SH	28.72	109.13	193.66	251.00	308.57	24.07	18.57	15.11
平均水平							18.33	15.24	12.87
维力医疗	603309.SH	11.74	1.92	2.21	2.70	3.23	15.48	12.67	10.58

USELESS

Original Output:

```
def solution():
    # Define variables name and value
    year_3_sales = 87.73
    year_5_sales = 150.01 (×)
    # Do math calculation to get the answer
    growth_rate = ((year_5_sales - year_3_sales) / year_3_sales)
* 100
    # return answer
    return round(growth_rate, 2)
    # Call the function and print the result
    print(solution())
```

Augmented Output:

```
def solution():
    # Define variables name and value
    year_3_sales = 87.73
    year_5_sales = 120.01
    # Do math calculation to get the answer
    growth_rate = ((year_5_sales - year_3_sales) / year_3_sales)
* 100
    answer = round(growth_rate, 2)
    # return answer
    return answer
```

Groud Truth: 36.79

Figure 6: Example of Filtering-Reasoning Pipeline

Question: Assuming the cash flow from financing activities according to U.S. GAAP is: Answer to a single decimal place.

Image:

Net income	$55,000
Depreciation	$60,000
Taxes paid	$25,000
Interest paid	$5,000
Dividends paid	$10,000
Cash received from sale of company building	$40,000
Issuance of preferred stock	$45,000
Repurchase of common stock	$20,000
Purchase of machinery	$20,000
Issuance of bonds	$40,000
Debt retired through issuance of common stock	$45,000
Paid off long-term bank borrowings	$10,000
Profit on sale of building	$15,000

Original Output:

```
def solution():
    # Define variables name and value
    lifo_reserve_2014 = 82  # in thousands
    tax_rate = 0.30  # 30%

    # Do math calculation to get the answer
    income_increase_before_tax = lifo_reserve_2014 (×)
    tax_on_increase = income_increase_before_tax * tax_rate
    income_increase_after_tax = income_increase_before_tax -
tax_on_increase

    # Return answer to one decimal place
    return round(income_increase_after_tax, 1)
```

Augmented Output:

```
def solution():
    # Define variables name and value
    lifo_reserve_current = 82  # LIFO reserve for 2014 (in
thousands)
    lifo_reserve_previous = 64  # LIFO reserve for 2013 (in
thousands)
    tax_rate = 0.30  # 30% corporate tax rate

    # Do math calculation to get the answer
    change_in_lifo_reserve = lifo_reserve_current -
lifo_reserve_previous
    pre_tax_income_increase = change_in_lifo_reserve
    additional_tax = pre_tax_income_increase * tax_rate
    net_income_increase = pre_tax_income_increase -
additional_tax
    answer = round(net_income_increase, 1)

    return answer
```

Error Analysis: The second implementation incorrectly assumes that the entire 2014 LIFO reserve ($82k) should be released and added to net income, resulting in an overstatement. In reality, only the change in LIFO reserve from 2013 to 2014 ($18k) impacts 2014 income when switching to FIFO. The first implementation correctly uses this difference and accounts for taxes, leading to an accurate calculation.

Ground Truth: 45000.0

Figure 7: Example of Knowledge Augmentation

Financial Function

```
def calc_net_return(init_investment: float,
     growth: float, fee_rate: float, inc_rate:
     float, hurdle: float) -> float:
    """
    Calculate the net return for an investor in
    a hedge fund given various parameters.

    Args:
    initial_investment (float): The initial
    amount invested in the hedge fund...

    Returns:
    net_return (float): The net return for the
    investor after fees, in millions.
    """
    end_value = init_investment * (1 + growth)
    fee = end_value * fee_rate
    net_value = end_value - fee
    hurdle_value = init_investment * (1 + hurdle
        )
    inc_fee = max(0, (net_value - hurdle_value)
    * inc_rate)
    net_return = end_value - (fee + inc_fee) -
    init_investment
    return round(net_return, 2)
```

Figure 8: Example of Retrieved Financial Function

Question: What is the anticipated portfolio return made by an investment company analyst named Maud, under two scenarios of portfolio returns in various economic conditions (expressed as a percentage) ? Answer to a single decimal place.

Image:

Scenario	Probability of scenario (%)	Portfolio return	Probability of return (%)
good economic situation	70	20%	50
good economic situation	70	10%	50
bad economic situation	30	5%	60
bad economic situation	30	-10%	40

Structured Data:

| Scenario | Probability of scenario (%) | Portfolio return | Probability of return (%) |\n| good economic situation | 70 | 20% | 50 |\n| good economic situation | 70 | 10% | 50 |\n| bad economic situation | 30 | 5% | 60 |\n| bad economic situation | 30 | -10% | 40 |

Figure 9: Example of Parsing-Reasoning Pipeline

Model	**Organization**	**Source**
Claude 3.7 Sonnet (Thinking)	Anthropic	claude-3-7-sonnet-20250219
Claude 3.7 Sonnet	Anthropic	claude-3-7-sonnet-20250219
GPT-4o	OpenAI	gpt-4o-2024-11-20
OpenAI o1	OpenAI	o1-2024-12-17
Gemini 2.0 Flash Thinking	Google DeepMind	gemini-2.0-flash-thinking-exp-01-21
Gemini 2.0 Pro	Google DeepMind	gemini-2.0-pro-exp-02-05
Gemini 2.0 Flash	Google DeepMind	gemini-2.0-flash
Gemma 3 27B	Google DeepMind	gemma-3-27b-it
Qwen-Omni-Turbo	Alibaba Qwen	qwen-omni-turbo-2025-01-19
Qwen2.5-VL-72B	Alibaba Qwen	qwen2.5-vl-72b-instruct
QvQ-72B-Preview	Alibaba Qwen	qvq-72b-preview
Pixtral Large	MistralAI	pixtral-large-latest
Mistral Small 3.1	Mistral AI	mistral-small-3.1-24b-instruct
InternVL2.5-78B	OpenGVLab	OpenGVLab/InternVL2_5-78B
Grok 2 Vision	xAI	grok-2-vision-1212
Llama 4 Maverick	AI@Meta	llama-4-maverick

Table 1: Specifications of MLLMs Evaluated on FinMMR.

Model	**Size**	**Extended thinking**	**Hard**		**Medium**		**Easy**		**Avg.**		**Token (M)**	
			CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT	CoT	PoT
Claude 3.7 Sonnet		✔ (64K)	**50.67**	**50.33**	66.00	63.00	**75.67**	**75.33**	**64.11**	**62.89**	**2.44**	**3.02**
Claude 3.7 Sonnet		✘	50.33	49.00	63.67	60.00	75.33	74.00	63.11	61.00	0.27	0.24
Qwen2.5-VL-72B	72B	✘	41.33	48.67	66.00	**64.00**	77.00	70.00	61.44	61.45	0.28	0.12
GPT-4o		✘	46.00	47.67	66.00	63.67	73.33	73.67	61.78	61.67	0.23	0.11
InternVL2.5-78B	78B	✘	38.00	47.67	57.00	60.67	69.00	69.33	54.67	59.22	–	–
Gemini 2.0 Flash Thinking		✔	46.67	46.67	**67.00**	60.33	73.67	73.33	62.45	59.00	0.33	0.13
Gemini 2.0 Flash		✘	46.67	46.00	63.33	55.67	70.00	71.33	60.00	57.67	0.31	0.12
Gemini 2.0 Pro		✘	45.00	44.00	61.33	62.00	71.33	71.33	59.22	59.78	0.23	0.12
OpenAI o1		✔	–	45.00	–	–	–	–	–	–	–	0.61
QVQ-72B-Preview	72B	✔	40.33	6.00	58.33	8.33	73.33	11.00	57.33	8.44	1.46	1.54
Qwen-Omni-Turbo		✘	18.33	30.33	35.67	45.33	53.67	60.67	35.89	45.44	0.24	0.11
Grok 2 Vision		✘	27.67	26.33	42.67	33.67	72.00	73.00	47.45	44.33	0.30	0.16
Pixtral Large	124B	✘	25.00	27.00	38.67	38.00	65.33	67.33	43.00	44.11	0.28	0.20

Table 2: Results of different models using CoT and PoT prompting methods on the *validation* set of FinMMR. We use the best Accuracy on the *Hard* subset as the ranking indicator of model performance.

CoT Prompt Template

```
SYSTEM_INPUT = '''You are a financial expert, you are supposed to answer the given
↪  question based on the provided image and context. You need to first think through
↪  the problem step by step, identifying the exact variables and values, and
↪  documenting each necessary step. Then you are required to conclude your response
↪  with the final answer in your last sentence as 'Therefore, the answer is {final
↪  answer}'. The final answer should be a numeric value.'''

USER_INPUT = '''The following question context is provided for your reference:
{question_context with image tags like <image1>}
Question:
{question_question}
Let's think step by step to answer the given question.'''
```

Figure 10: CoT Prompt Template

PoT Prompt Template

````
SYSTEM_INPUT = '''You are a financial expert, you are supposed to generate a Python
↪  program to answer the given question based on the provided image and context. The
↪  returned value of the program is supposed to be the answer. Here is an example of
↪  the Python program:
```python
def solution():
    # Define variables name and value
    revenue = 600000
    avg_account_receivable = 50000

    # Do math calculation to get the answer
    receivables_turnover = revenue / avg_account_receivable
    answer = 365 / receivables_turnover

    # return answer
    return answer
```
'''

USER_INPUT = '''The following question context is provided for your reference:
{question_context with tags like <image1>}
Question:
{question_question}
Please generate a Python program to answer the given question. The format of the
↪  program should be the following:
```python
def solution():
    # Define variables name and value

    # Do math calculation to get the answer

    # return answer
```

Continue your output:
```python
def solution():
    # Define variables name and value
```
'''
````

Figure 11: PoT Prompt Template

RAG Prompt Template

```
SYSTEM_INPUT = '''You are a financial expert, you are supposed to answer the given
↪  question. You need to first think through the problem step by step, identifying
↪  the exact variables and values, and documenting each necessary step. Then you are
↪  required to conclude your response with the final answer in your last sentence as
↪  'Therefore, the answer is {final answer}'. The final answer should be a numeric
↪  value.'''

USER_INPUT = '''The following question context is provided for your reference:
{question_context with image tags like <image1>}
Question:
{question_question}

To assist you in solving the problem, I will provide some financial functions for
↪  reference.
Important Notes:
1. Do not directly call or use the provided reference functions in your
↪  implementation.
2. The reference functions are only intended to help you understand the logic and
↪  approach for solving the problem, when it is appropriate.
3. You must implement the solution from scratch, using your own code and logic.
4. Ensure that your implementation is independent and does not rely on the provided
↪  functions.
5. If the reference functions are not applicable to the problem, ignore them entirely
↪  and focus on solving the problem based on your own understanding.

The following are the financial functions for your reference:
{financial_function}'''
```

Figure 12: RAG Prompt Template

Function Relevance Judgment Prompt Template

```
SYSTEM_INPUT = '''You are a financial expert, you are supposed to judge whether the
↪  given financial function is useful for answering the question.

For each function, follow these guidelines:
1. Determine if the function can directly address the user's problem, considering the
↪  function's purpose, input parameters, and return values.
2. Consider the applicability range of the function, assumptions, limitations, and
↪  restrictions when evaluating if it's relevant.
3. If the function can effectively contribute to solving the problem or is essential
↪  for the calculation or analysis required, respond with [USEFUL].
4. If the function cannot effectively help in solving the problem, or is irrelevant
↪  based on its scope and assumptions, respond with [USELESS].

Use financial domain knowledge to ensure that each judgment is precise and aligned
↪  with common practices for problem-solving in the finance domain.'''

USER_INPUT = '''Given a financial question and financial functions, I want you to
↪  analyze each of these function to assess if it can be useful in solving the
↪  question.

For each financial function:
1. You need to decide if it is useful based on its fit with the problem's requirements
↪  and constraints.
2. If the function is relevant to solving my problem, output [USEFUL].
3. If it is not helpful, output [USELESS].
Do not include any additional explanation, just the relevant outputs for each
↪  function.

Question:
{financial_question}

Functions:
{financial_function}

Output the results in the following format:
[USEFUL, USELESS, USELESS, ...]'''
```

Figure 13: Function Relevance Judgment Prompt Template

Visual Relevance Judgment Prompt Template

```
SYSTEM_INPUT = '''Given a complex financial numeric reasoning question and a set of
↪  images, analyze each image and decide if it is useful for solving the question.
Guidelines:
1. Determine the content displayed and described in each image. Determine whether the
↪  image contains data or information that is used (directly or indirectly) in the
↪  multi-step calculations leading to the answer.
2. If the image's content does not contribute to the solution, output [USELESS].
3. In cases of uncertainty, lean towards labeling an image as [USEFUL] rather than
↪  [USELESS]. The question is designed such that at least one image contains
↪  necessary data. If you think none are clearly helpful, choose the one image most
↪  likely to contain relevant data and label it [USEFUL].
4. You need to first think through the problem step by step, documenting each
↪  necessary step. Then you are required to conclude your response with the final
↪  answer in your last sentence as 'Therefore, the answer is {final answer}'. The
↪  final answer should be the labels in the order of the images. For example:[USEFUL,
↪  USELESS, USELESS, ...]
Example: For such a question that "Based on the data in the image, calculate the
↪  growth rate of bauxite imports from 2021 to 2022, and round the result to two
↪  decimal places."For this problem, you may infer that in order to solve it, the
↪  import amounts for 2021 and 2022 must be obtained from the images. Once you
↪  identify an image that contains the relevant data, label that image as [USEFUL].
You may encounter other styles of charts such as bar charts, line charts, etc. with
↪  clear data. Regardless of the category of the image, please carefully consider and
↪  analyze the data in the image according to the above rules.'''

USER_INPUT = '''Given a complex financial numeric reasoning question and a set of
↪  images, analyze each image and decide if it is useful for solving the question.
Guidelines:
1. Determine whether the image contains data or information that is used (directly or
↪  indirectly) in the multi-step calculations leading to the answer.
2. If the image's content does not contribute to the solution, output [USELESS].
3. In cases of uncertainty, lean towards labeling an image as [USEFUL] rather than
↪  [USELESS]. The question is designed such that at least one image contains
↪  necessary data. If you think none are clearly helpful, choose the one image most
↪  likely to contain relevant data and label it [USEFUL].
4. You need to first think through the problem step by step,  documenting each
↪  necessary step. Then you are required to conclude your response with the final
↪  answer in your last sentence as 'Therefore, the answer is {final answer}'. The
↪  final answer should be the labels in the order of the images. For example:
↪  [USEFUL, USELESS, USELESS, ...]
Example: For such a question that "Based on the data in the image, calculate the
↪  growth rate of bauxite imports from 2021 to 2022, and round the result to two
↪  decimal places." For this problem, you may infer that in order to solve it, the
↪  import amounts for 2021 and 2022 must be obtained from the images. Once you
↪  identify an image that contains the relevant data, label that image as [USEFUL].
↪  You may encounter other styles of charts such as bar charts, line charts, etc.
↪  with clear data. Regardless of the category of the image, please carefully
↪  consider and analyze the data in the image according to the above rules. Let's
↪  think step by step to answer the given question.
Question:
{financial_question}
'''
```

Figure 14: Visual Relevance Judgment Prompt Template