Evaluating Mathematical Reasoning in In-Context Learning

Cindy Luo kl3108@nyu.edu
Shulin Ji sj4016@nyu.edu
Jiayuan Song js10417@nyu.edu
Vivian Yan qy620@nyu.edu

Abstract

This research examines the adaptability of Large Language Models (LLMs) in in-context learning, with a particular emphasis on mathematical reasoning involving both standard and novel notations. We explored whether LLMs can transcend pattern recognition to interpret and utilize unfamiliar symbols, linking them back to the common operations. To answer this question, we conducted a comprehensive evaluation with varying prompting methods and mathematical operations across models with different scales. Our findings reveal a significant and consistent decline in accuracy when LLMs encounter novel symbols in different prompting paradigms, underscoring the challenges they face with unfamiliar expressions.

1 Introduction

Recent advances in Large Language Models (LLMs) have sparked considerable interest in their capabilities, particularly in the realm of in-context learning [1, 2]. Previous research has suggested that high-level, multi-task broad cognitive capabilities accquired by the LLMs may be extracted by simple prompting techniques [3], and various prompting methods have demonstrated substantial improvements in performances on areas such as symbolic and mathematical reasoning tasks [4, 5]. However, while evaluating the learning outcomes in mathematical reasoning, prior research predominantly utilized conventional mathematical expressions to probe these models [6, 7, 8], leaving us with an open question: can LLMs demonstrate similar adaptability and understanding when encountering unfamiliar symbols or novel contexts?

To address this gap, our research investigated whether LLMs can go beyond mere pattern matching to comprehend and utilize unfamiliar symbols in mathematical operations. For instance, when presented with a novel operator (such as '@' in our experimental setup), can these models effectively map it to known mathematical operations based on a few contextual examples? This question is crucial for evaluating the depth of LLMs' understanding and their ability to apply learned knowledge in new and diverse contexts.

Our study revealed that across all scales, models showed reduced performance with novel operators in prompts, irrespective of the prompting method used. Our findings indicate that existing prompting methods might be less adept at teaching 'new' operations in in-context learning, suggesting a need for future research to explore learning outcomes with the introduction of uncommon notations.

2 Related Work

Zhou et al. (2022) [7] introduced a novel prompting paradigm, *algorithmic learning*, designed to enhance a model's ability to understand and apply algorithmic principles in mathematical reasoning. To evaluate the efficacy of this method, they compared it with three established prompting methods: Few-Shot [2], Chain of Thought [1], and Instruction-only [7], across various mathematical tasks. Their findings demonstrated that *algorithmic learning* significantly boosts learning efficiency, particularly

in out-of-distribution calculations, by improving a model's ability to parse and assimilate reasoning steps. Inspired by their approach and evaluation framework, our study has adopted a similar testing methodology to further explore these dynamics in the context of our research paradigm by introducing the novel-vs-common-notation task.

3 Approach

3.1 Operations and Masking

We decided to evaluate the model's mathematical reasoning ability in multiple operations. We selected three common arithmetic operations: (1) two 3-digits addition; (2) two 3-digits subtraction; and (3) 3-digit-by-1-digit multiplication.

For the use of novel notations in evaluations, we introduced the unfamiliar $@$, $\#$, and $\$$ in prompts in replacement of the $+$, $-$, and $*$ for the three operations above.

3.2 Prompting Methods

In **Few-shot** prompts (Appendix E.1), we provided many pairs of question and correct answer.

In **Chain-of-Thought** prompts (Appendix E.2), we included a step-by-step thought process for each example question in addition to the correct answer, which involves breaking down the second number by digit before operations. For multiplication, the thought process is based on the distributive law.

We designed **Instruction-only** prompts (Appendix E.3) that encompassed clear definitions of the three operations with abstract variables and step-by-step explanations of the mathematical rules , in which a crucial consideration was the masking of literal operations. We decided not to mask tokens like *'add'/'sum'* to remain an intact context of instruction. Otherwise, replacements without providing any instructional rules of $@$, for example, would violate the objective of instruction-only prompts, and the model would lack comprehension, as it had not been trained on the concept of *'apply* $@$ *operation'*. Instead, by substituting math operators only while preserving the instructional context, we struck a balance between maintaining the original format (unchanged) and adopting a fully-masked format (complete substitution). This intermediate state served as a suggestive compromise, aligning with our objective of exploring the model's learning tendencies.

Our **Algorithmic** prompts (Appendix E.4) were partially adopted from [7], in which every number including the answer is represented as a named array, and each digit is indexed during operations. We verified its necessity, which enforce a concept of location. Each illustration, looping from the last to the first digit, performs digit-to-digit operation and prepends the answer digit to an array.

Since the algorithm involves multiple steps, where errors can be introduced from any, we explored a list of optimizations and ablations to the original prompts, results shown in Appendix D.5, including

- the replacement of 'a % 10' by a literal 'last digit' to lessen mod mistakes.
- showing a prepending of 0 to the answer array at the first position if the carry is 0.
- the replacement of "has x digits" by "has length x" to enforce a length concept for output.
- removing the declarations of calculation end and performance end to shorten prompt length.
- the use of literal instructions 'prepend' and 'append' instead of showing the resulting array.
- removing logical connectives like 'thus' from the prompts.

4 Experiments

We used LLaMa-2 as our model for experiments, which is a large language model developed by Meta and comes in several sizes. We leveraged the API from Together.ai, a cloud platform supporting the implementation of generative AI, for interactions with LLaMa-2.

4.1 Methodology

We performed experiments with models of different scales: LLaMa-2-7B, LLaMa-2-13B, and LLaMa-2-70B. The configuration for different prompting methods were tuned and included in Appendix C.

We performed 4 trials, each with 30 queries, on each of the three mathematical operations (addition, subtraction, and multiplication) and their corresponding masked symbolic operations. We utilized the same random seed for question generated in each trial across different prompting methods to ensure that models were evaluated using the same question set.

For few-shots and chain-of-thought prompting, we first used 10 examples for each of the 6 operations. For instruction prompting, there are no explicit examples. For algorithmic prompting, we only generated 5 examples due to length limitations. Since we manually provided the prompts in the string format, there was no additional preprocessing. We extracted the numerical final results of each problem using regular expression in post-processing, which differed by prompts.

4.2 Evaluation

We evaluated the model performance by comparing the average accuracy and average mean absolute error (MAE) of all four trials. The evaluation includes three aspects: (1) the performance of a specific language model (e.g. LLaMa-2-7B) with different prompting methods; (2) the performance differences between prompting with existed symbols and with masked ones, suggesting whether language models can identify the symbol in a masked math problem and solve it reasonably; (3) whether such performance differences change if model becomes larger (7B v.s. 13B v.s. 70B).

4.3 Results

Figure 1 showcases the mean performance of all experiments, with the complete record in Appendix D.

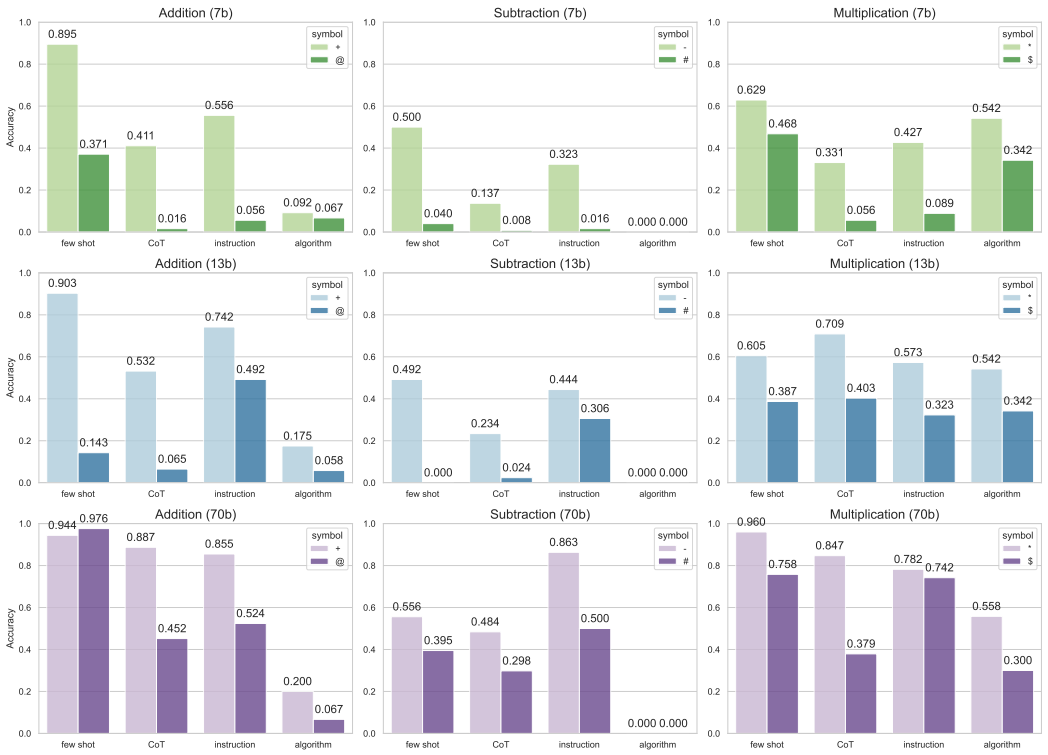

Figure 1: Performances of all prompts, all operators and all model scales

Our experimentation across various operations, models of different scales, and prompting methods revealed that the accuracy with masked operators was consistently lower than that in unmasked conditions. Specifically, we noted that:

• By comparing the figure's mid column, representing the model's performance on the subtraction task, with columns on the sides, we could observe that accuracy drops were the largest in subtraction,

followed by addition, and the least in multiplication.
• Larger-scale models exhibited a smaller drop in performance. For example, there is consistent performance increase comparing the last row (70B) to the first row (7B).
• For smaller models, the accuracy gap between masked and unmasked conditions was the smallest under few-shot prompting (the left-most bars of the first row). As the model scale increased, complex prompting techniques, like instructional prompts (the third set of bars), also proved effective in narrowing this gap. Chain of Thought (CoT) did not adapt well to learning novel symbols.

When comparing across operations, addition tasks were executed with greater ease than multiplication, which in turn was simpler than subtraction. Additionally, larger models consistently outperformed their smaller counterparts in both masked and unmasked conditions, indicating a correlation between model size and mathematical problem-solving ability.

5 Analysis

Comparison of Masked and Unmasked Conditions The masked task presented a unique challenge, highlighting several key abilities necessary for successful task completion. These include the capability for basic arithmetic operations (addition, subtraction, multiplication), the recognition of symbols as abstract representations of operations, and the extraction and linkage of these abstract operations from the symbols. Crucially, models needed to be open to the possibility that different notations could define identical operations and capable of mapping novel notations to these abstract operations. Any failure in these steps, whether in in-context learning or the model's original training capacity, could potentially lead to a performance drop. We have shown that even after in-context learning, models struggle with novel representations that deviate from their training data.

Performance Drop Across Operations Our analysis revealed partial alignment with human performance patterns [9], particularly with addition being the easiest task, likely due to its higher baseline in the unmasked version. However, it was intriguing to find that multiplication had a lower accuracy gap. This could be attributed to the commonality of multiple symbols representing multiplication, such as '*', '×', and '·'. Therefore, if a model recognizes that different notations can represent the same operation, the unmasked version becomes less challenging. Subtraction's increased difficulty could be related to the introduction of negative results in the questions.

Prompting Methods and Model Scale Larger models possibly developed more advanced meta-cognitive skills, as evidenced by their smaller accuracy gap. Few-shot prompting, which offers more examples, might facilitate pattern matching from the examples provided in the prompts, making it more effective for smaller models. In contrast, larger models seemed to benefit more from detailed, cognitively demanding prompts, as observed by the closing accuracy gap. CoT, however, did not prove effective, potentially because it does not provide room for mere pattern matching in prompts or offer detailed explanations about the new symbol. The prompts of CoT also have the highest number of novel symbols as compared to those of the other methods, adding to the confusion of the model.

6 Conclusion

Our research findings reveal intriguing contrasts in model performances when adopting common versus novel notations in mathematical tasks. The use of novel symbols in place of common notation resulted in a noticeable decline in accuracy and an increase in MAE, highlighting challenges LLMs face with unfamiliar representations. In terms of mathematical operations, the models' performance mirrored human capabilities, with the easiest task being three-digit addition, followed by three-to-one digit multiplication, and the most challenging being three-digit subtraction. Furthermore, the study indicates that larger models derive greater benefits from detailed, cognitively demanding prompts, such as algorithmic and instructional types, aligning with the increased complexity and sophistication of these models. Additionally, larger models exhibited a smaller decline in performance when using novel symbols, suggesting that meta-cognitive skills emerged from models with larger scales.

A key limitation of our study stems from the difficulty in standardizing model configurations across diverse prompting techniques, which may have led to the less promising results of our adapted prompting paradigm compared to prior studies. Future research should delve into the observed varying performances across different mathematical operations and examine how the number of examples in prompts affects performance, especially under both unmasked and masked conditions.

A Acknowledgments

This work was part of the course work for the DS-GA 1011 Natural Language Processing with Representation Learning course in Fall 2023. It was supported by the instructors of the course, as well as the Center for Data Science at New York University.

B Contribution

Cindy Luo contributed to few-shot prompting, analysis and literature review. Vivian Yan contributed to Chain-of-Thought prompting. Shulin Ji contributed to designing instruction-only prompts. Jiayuan Song specialized in experiments and designs about the Algorithmic approach. Everyone contributed to writing the report.

C Model parameters

Table 1: Config Parameters

	max_tokens	temperature	top_k	top_p	repetition_penalty	stop
Few-shot	70	0.2	50	0.7	1	[]
Chain-of-thought	70	0.2	50	0.7	1	[]
Instruction-only	200	0.7	50	0.7	1	[]
Algorithmic	700	0.7	50	0.6	1	[]

D Detailed Results

D.1 Few Shots

Few-shots: Accuracy						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	0.895	0.371	0.5	0.040	0.629	0.468
LLaMa-2-13B	0.903	0.143	0.492	0.0	0.605	0.387
LLaMa-2-70B	0.944	0.976	0.556	0.395	0.960	0.758

Few-shots: MAE						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	6.33	161.44	257.27	341.81	44.73	123.48
LLaMa-2-13B	8.81	256.10	254.52	357.44	22.43	805.01
LLaMa-2-70B	3.01	0.826	226.43	280.40	5.85	191.45

D.2 Chain-of-Thought

Chain-of-Thought: Accuracy						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	0.411	0.016	0.137	0.008	0.331	0.056
LLaMa-2-13B	0.532	0.065	0.234	0.024	0.709	0.403
LLaMa-2-70B	0.887	0.452	0.484	0.298	0.847	0.379

Chain-of-Thought: MAE						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	50.83	238.63	290.70	522.13	106.29	1282.65
LLaMa-2-13B	28.11	365.49	235.30	365.93	72.21	297.18
LLaMa-2-70B	95.07	251.82	240.65	226.53	71.53	1065.15

D.3 Instruction-only

Insturction-only: Accuracy						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	0.556	0.056	0.323	0.016	0.427	0.089
LLaMa-2-13B	0.742	0.492	0.444	0.306	0.573	0.323
LLaMa-2-70B	0.855	0.524	0.863	0.500	0.782	0.742

Insturction-only: MAE						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	1069.28	6997.53	299.76	486.91	567.34	3009.01
LLaMa-2-13B	3287.91	10114.82	239.84	271.88	784.52	1437.71
LLaMa-2-70B	94.82	309.27	36.78	228.61	7350.75	915.41

D.4 Algorithmic

Insturction-only: Accuracy						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	0.092	0.067	0.0	0.0	0.042	0.0
LLaMa-2-13B	0.175	0.058	0.0	0.0	0.542	0.342
LLaMa-2-70B	0.2	0.067	0.0	0.0	0.558	0.3

Insturction-only: MAE						
Model	Addition		Subtraction		Multiplication	
	+	@	-	#	*	$
LLaMa-2-7B	1506.99	1002.14	945.45	7706.58	2651.52	2668.35
LLaMa-2-13B	808.38	480.63	2343.41	3478.82	226.68	585.21
LLaMa-2-70B	855.9	1397.34	693.08	4943.08	538.12	194.92

D.5 Optimization for Algorithmic

The optimization was performed on addition with the 70b model. The accuracy with the original prompt was 0.075. After replacing the mathematical 'mod' operation with the literal instruction and forcing a prepending of 0 improved the accuracy 100% to 0.15. After replacing 'digits' with length, the accuracy was improved to 0.199, which bumped to 0.20 by excluding the declarations of finished calculation that also shortened the prompt length.

Experiments with words like 'prepend' and 'append' had a lower performance of 0.19, during which many *swaps of answer digits* were observed, showcasing the model's lack of knowledge on the location. Removing logical connectives decreased the accuracy to 0.12, implying the significance of them completing a thought process and the necessity of 'teaching' a logic in prompt.

E Prompt Examples

The following examples only contain one illustration. Many of these are provided during experiments.

E.1 Few Shots

	Unmasked	Masked
Addition	Q: 128+367=? A:495	Q: 128@367=? A:495
Subtraction	Q: 723-850=? A:-127	Q: 723#850=? A:-127
Multiplication	Q: 374*6=? A:2244	Q: 374$6=? A:2244

E.2 Chain-of-Thought

	Unmasked	Masked
Addition	Problem: 128+367=? Explanation: Let's think step by step. 128+ 367= 128+ 300+ 60+ 7= 428+ 60+ 7= 488+ 7= 495. Answer:495	Q: 128@367=? Explanation: Let's think step by step. 128@ 367= 128@ 300@ 60@ 7= 428@ 60@ 7= 488@ 7= 495. Answer:495
Subtraction	Problem: 723-850=? Explanation: Let's think step by step. 723- 800- 50= -77- 50= -127. Answer: -127	Problem: 723#850=? Explanation: Let's think step by step. 723# 800# 50= -77# 50= -127. Answer: -127
Multiplication	Problem: 374*6=? Explanation: Let's think step by step. 374*6=(300+70+4)*6=300*6+ 70*6+4*6=1800+420+24=2244. Answer: 2244	Problem: 374$6=? Explanation: Let's think step by step. 374$6=(300+70+4)$6=300$6+ 70$6+4$6=1800+420+24=2244. Answer: 2244

E.3 Instruction-only

For addition:

```
The following are instructions for solving addition problems in the
    form of x + y = z, where x, y, and z are positive integers. We
    will use the standard algorithm for addition. We align the numbers
     x and y on the least significant digit, which is the ones digit.
    Starting from right to left, we go from the least significant
    digit to the most significant digit and add the corresponding
    digits from each number. When the sum of the two digits is greater
     than 9, a carry of 1 is included in the sum of the next digits.
    When there is only one digit available from the two numbers, only
    that digit along with any carry is included in the sum. When all
    the digits are processed, only the remaining carry if any shall be
     included in the sum.
For x + y = ? where x = int(str(abc)), y = int(str(efg)), we can solve
     it with the following steps:
1. Begin with the rightmost digits (c and g).
2. Add these digits together, and write down the last digit of the
    result.
3. Move to the next digits (b and f), add them along with any carry
    from the previous step.
4. Repeat this process for the leftmost digits (a and e).
5. If there's a carry after adding the leftmost digits, include it in
    the final answer.
Combine all the results to obtain the sum.
```

```
The answer should be in the form below:
Question: What is abc+efg=?
Answer:
 abc
+efg
-----
 hij
The answer is hij.
Question: What is
```

Masked addition is to substitute the + symbol with the @ symbol.

For subtractions, we change the above concept of 'carry' to 'borrow', and introduce the concept of negative:

```
...
For x - y = ? where x = int(str(abc)), y = int(str(efg)), we can solve
     it with the following steps:
    1. Begin with the rightmost digits (c and g).
        a. If c is greater than or equal to g, subtract g from c and
            write down the result.
        b. If c is less than g, borrow 1 from the next higher-order
            digit and subtract g from (c + 10).
    2. Move to the next digits (b and f).
        a. Subtract f from b along with any borrow from the previous
            step.
        b. If b is less than f, borrow 1 from the next higher-order
            digit.
    3. Move to the leftmost digits (a and e).
        a. Subtract e from a along with any borrow from the previous
            step.
        b. If a is less than e, borrow 1 from the next higher-order
            digit.
    4. Combine all the results to obtain the difference.
    5. If there's a borrow after subtracting the leftmost digits,
        include a negative mark '-' in the final answer at the
        leftmost place.
...
```

Masked subtractions is to substitute the − symbol with the # symbol.

For multiplication, we preserve the above concept of 'carry', and change the operated digits in each step:

```
...
For the multiplication x * y, where x is a 3-digit number (abc) and y
    is a 1-digit number (d), follow these steps:
    1. Begin with the rightmost digit of the multiplier (c).
        a. Multiply it by d and write down the result. Record any
            carry.
    2. Move to the next digit of the multiplier (b).
        a. Multiply it by d, add the previous carry, and write down
            the result. Record any new carry.
    3. Move to the leftmost digit of the multiplier (a).
        a. Multiply it by d, add the previous carry, and write down
            the result.
    4. If there is still a carry after processing the leftmost digit,
        include it in the final result.
    5. The final result is the concatenation of the results obtained
        in each step.
...
```

Masked multiplication is to substitute the ∗ symbol with the $ symbol.

E.4 Algorithmic

Addition:

```
Problem: 456 + 789=
Explanation:
The first number is 456, FN=[4, 5, 6]. The second number is 789, SN
    =[7, 8, 9]. Length of FN, [4, 5, 6], is 3. Length of SN, [7, 8,
    9], is 3. thus the maximum number of digits is 4.
In each subsequent step, we remove one number from the end of FN and
    one from the end of SN.
Length of FN is 3. FN=[4, 5, 6]. Length of SN is 3. SN=[7, 8, 9]. FN
    [3]=6. SN[3]=9. C[3]=0. 6+9+0=15. 15 last digit is 5. Length of A
    is 1. Thus A=[5]. Since (15-5)/10=1 C[2]=1.
Length of FN is 2. FN=[4, 5]. Length of SN is 2. SN=[7, 8]. FN[2]=5.
    SN[2]=8. C[2]=1. 5+8+1=14. 14 last digit is 4. Length of A is 2.
    Thus A=[4, 5]. Since (14-4)/10=1, C[1]=1.
Length of FN is 1. FN=[4]. Length of SN is 1. SN=[7]. FN[1]=4. SN
    [1]=7. C[1]=1. 4+7+1=12. 12 last digit is 2. Length of A is 3.
    Thus A=[2, 4, 5]. Since (12-2)/10=1, C[0]=1.
There are no more digits. Length of A is 4. C[0]=1. Thus A=[1, 2, 4,
    5]. The final Answer is [1, 2, 4, 5].
```

Masked addition is to substitute the + symbol with the @ symbol.

Subtraction:

```
Problem: 29-570=
Explanation:
The first number is 29, adding commas between each number, FN=[2,9].
    The second number is -570, adding commas between each number, SN
    =-[5,7,0]. FN [2,9] has 2 digits, SN -[5,7,0] has 3 digits, max is
     3.
Len(FN)=2. FN=[2,9]. FN[2]=9. Len(SN)=3. SN=-[5,7,0]. SN[3]=-0. C
    [3]=0. Since 9-0+0=9, 9<10, 9%10=9. Len(A)=1. A=[9]. Since (9-9)
    /10=0, C[2]=0.
Len(FN)=1. FN=[2]. FN[1]=2. Len(SN)=2. SN=-[5,7]. SN[2]=-7. C[2]=0.
    Since 2-7+0=-5, -5<-10, -5%-10=-5. Len(A)=2. A=[-5,9]. Since
    (-5--5)/10=0, C[1]=0.
Len(FN)=0. FN=[]. FN[0]=empty. Len(SN)=1. SN=-[5]. SN[1]=-5. C[1]=0.
    Since 0-5+0=-5, -5<-10, -5%-10=-5. Len(A)=3. A=[-5,-5,9]. Since
    (-5--5)/10=0, C[0]=0.
Len(FN)=0. FN=[]. FN[0]=empty. Len(SN)=0. SN=-[]. SN[0]=empty. Since
    both FN and SN are empty, next. Since C[0]=0, the steps are done.
    Since there are - in A, we check the sign of
the last step A[1]=-5. Since -5 is neg, we change the sign and process
     A from right to left. A=[-5,-5,9]=-[+5,+5,-9]. C[3]=0.
Len(A)=3. A=-[+5,+5,-9]. A[3]=-9. Since -9<0, B=10, C[2]=-1. Since C
    [3]=0, thus -9+10+0=1. Len(ANEW)=1. ANEW=-[1]. C[2]=-1.
Len(A)=2. A=-[+5,+5]. A[2]=+5. Since +5>0, B=0, C[1]=0. Since C[2]=-1,
     thus 5+0-1=4. Len(ANEW)=2. ANEW=-[4,1]. C[1]=0.
Len(A)=1. A=-[+5]. A[1]=+5. Since +5>0, B=0, C[0]=0. Since C[1]=0,
    thus 5+0+0=5.
Len(ANEW)=3. ANEW=-[5,4,1]. C[0]=0.
Len(A)=0. A=-[]. Since A is empty, the problem is complete. The final
    Answer is -[5,4,1].
```

Masked subtractions is to substitute the − symbol with the # symbol.

Multiplication:

```
Problem: 579*6=
Explanation:
The first number is 579, FN=[5, 7, 9]. The second number is 6, SN=[6].
     Since FN, [5, 7, 9], has 3 digits, SN, [6], has 1 digit, thus the
     maximum number of digits is 4.
```

```
Length of FN is 3. FN=[5, 7, 9]. FN[3]=9. SN=6. C[3]=0. 9*6+0=54. 54
    last digit is 4. Length of A is 1. Thus A=[4]. Since (54 - 4) /
    10=5, C[2]=5.
Length of FN is 2. FN=[5, 7]. FN[2]=7. SN=6. C[2]=5. 7*6+5=47. 47 last
     digit is 7. Length of A is 2. Thus A=[7, 4]. Since (47 - 7) /
    10=4, C[1]=4.
Length of FN is 1. FN=[5]. FN[1]=5. SN=6. C[1]=4. 5*6+4=34. 34 last
    digit is 4. Length of A is 3. Thus A=[4, 7, 4]. Since (34 - 4) /
    10=3, C[0]=3.
There are no more digits, and C[0]=3. Thus the process is complete.
    Since there are no more operators, the final answer is [3, 4, 7,
    4].
```

Masked multiplication is to substitute the $*$ symbol with the \$ symbol.

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[4] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022.

[5] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2023.

[6] Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models, 2023.

[7] Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. Teaching algorithmic reasoning via in-context learning, 2022.

[8] Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schölkopf, and Mrinmaya Sachan. A causal framework to quantify the robustness of mathematical reasoning with language models, 2023.

[9] Miriam Rosenberg-Lee, Ting-Ting Chang, Chelsea B Young, Susan Wu, and Vinod Menon. Functional dissociations between four basic arithmetic operations in the human posterior parietal cortex: a cytoarchitectonic mapping study. *Neuropsychologia*, 49(9):2592–2608, 2011.