

# Hint of Thought prompting: an explainable and zero-shot approach to reasoning tasks with LLMs

Anonymous ACL submission

## Abstract

As a way of communicating with users and any LLMs like GPT or PaLM2, prompting becomes an increasingly important research topic for better utilization of LLMs. Although simple prompting performs well on single-step questions, it cannot permanently activate the correct knowledge path for multi-step reasoning tasks. The chain of thought (CoT), which often contains zero-shot CoT and few-shot CoT, is a recently developed prompting method that can explain the reasoning process to the LLM and outperforms simple prompting in three challenging reasoning tasks, including arithmetic, symbolic, and commonsense reasoning. In this paper, we propose a novel hint of thought (HoT) prompting with explainability and zero-shot generalization. First, it is decomposed into the following three steps: explainable sub-questions, logical reasoning, and answer extraction. Second, such three steps are sequentially ordered in the format of step-by-step hints, which can be easily adjusted and explained to different tasks. Finally, experimental results demonstrate that our HoT prompting has a significant advantage on the zero-shot reasoning task compared to existing zero-shot CoT. We did zero-shot experiments on math tasks like GSM8K, ADDSUB, AQUA, SVAMP and commonsense tasks such as StrategyQA. In particular, the accuracy of the proposed HoT prompting is improved with GSM8K from 40.50% to 67.80%, with AQUA from 31.9% to 46.4%, with SVAMP from 63.7% to 76.9%, and with ADDSUB from 74.7% to 87.34%, respectively, which even defeats the competitive PoT approach on GSM8k, AQUA, and SVAMP.

## 1 Introduction

Many researchers indicate that scaling up the size of generative language models and training datasets plays a critical role in recent NLP research. There are some powerful examples of large language models (LLMs), such as ChatGPT (?), PaLM (Chowdhery et al., 2022), and LAMDA (Thoppilan et al.,

2022). Indicated by the robustness of GPT-4 (OpenAI, 2023), many valuable applications can be developed, and (Bubeck et al., 2023) believes that it has the potential to lead the world of artificial intelligence (AI) to the world of artificial general intelligence (AGI).

The success of LLMs is often related to zero-shot or few-shot learning (in-context learning). This attribute helps the model to understand and solve different tasks by sampling a few examples (in-context learning) or only providing the instructions (zero-shot). The prompt engineering (Liu et al., 2021), or prompting, is a method for the users to interact with LLMs. To implement this method, researchers either design prompts manually (Schick and Schütze, 2021; Reynolds and McDonell, 2021) or generate them automatically (Gao et al., 2021; Shin et al., 2020) per task. Recently, prompt engineering has become a hot topic in NLP.

Although there is a fantastic performance that LLMs can solve single-step or intuitive tasks very well with task-specific zero-shot prompting or in-context learning prompting (Liu et al., 2021), models with scale up to 100B+ parameters are struggling to solve multi-step reasoning tasks (Rae et al., 2022). To address this drawback, researchers have introduced a method called chain of thought (CoT) prompting (Wei et al., 2022a), rather than standard question and answer examples prompting, and it feeds LLMs with a chain of reasoning examples (example). In other words, CoT demonstrates a reasoning path that composes an original complex question into multiple more straightforward steps. With CoT, the performance of language models on a large scale has dramatically improved. For example, a giant PaLM with 540B parameter (Chowdhery et al., 2022) with CoT can increase the accuracy of math calculations.

A few-shot prompting with CoT (Wei et al., 2022a) or without CoT (Brown et al., 2020) can finish many task-specific prompting tasks. Decom-

044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
054  
055  
056  
057  
058  
059  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
080  
081  
082  
083  
084

posed Prompting (Khot et al., 2023) proposed a model to decompose the task into sub-tasks and solve them iterative; however, it requires few-shot prompts and has to prompt a few times to get the result. To do zero-shot reasoning, researchers proposed zero-shot CoT, showing that LLMs are good zero-shot reasoners by simply adding "let's think step by step" (Kojima et al., 2022). However, the solving process is provided by the LLM without any explanation between each step. Also, Program of Thought (PoT) (Chen et al., 2023) performed better using Python as an extended tool for zero-shot math reasoning but mainly focused on math reasoning tasks. Both works show the difficulty of reasoning tasks with LLMs. To build an explainable, logical, and user-friendly end-to-end zero-shot prompt method, we propose hint of thought (HoT) from another point of view. To approach the solution, we take the traditional zero-shot CoT as an activation path that activates the correct reasoning path within LLMs. We propose a hints chain that is explainable to humans and efficient for the LLM to understand. It contains three main steps (1) **ask for five subquestions;** (2) **ask pseudocode as an answer for subquestions;** (3) **get the answer in a wanted way.** The hints chain automatically leads the LLM to do zero-shot reasoning by asking small sub-questions and answering sub-questions in pseudocode. The sub-questions can help users understand the "mind map" of the LLM, and pseudocodes provide a transparent, logical reasoning process. Moreover, answering in pseudocode can avoid semantic ambiguity. We used GPT-3.5-turbo (Brown et al., 2020) as our LLM; it is designed to give a complete sentence as an answer (Ye et al., 2023).

## 2 Background

### 2.1 Large language models and prompting

A language model (LM) is a model that is designed for estimating the probability distribution of text. In recent research, they found that scaling up the model size can help improve the performance (from a few million (Merity et al., 2016) to hundreds of millions (Devlin et al., 2019) to hundreds of billions (Brown et al., 2020) parameters). And the training data also becomes more extensive, e.g., webtest corpora (Gao et al., 2020). These improve the abilities of pre-trained LLMs in many downstream NLP tasks. Unlike the classic paradigm of "pre-train and fine-tune," an LLM that scales to

100B+ parameters displays the ability to few-shot learning (Brown et al., 2020). Using in-context learning, we can use prompts to strongly lead the generation to output a desired answer to a specific task. It is called pre-train and prompts (Liu et al., 2021).

### 2.2 Zero-shot CoT (Kojima et al., 2022)

Based on the drawback of few-shot CoT that costs time and people to design the prompt, (Kojima et al., 2022) proposed a zero-shot CoT prompt. They added "Let's think step by step" or a similar text. This work showed that the potential of LLMs is zero-shot. Compared with standard zero-shot and zero-shot CoT, the latter significantly improved with GSM8K from 12.5% to 40.5%.

### 2.3 Program of Thought (Chen et al., 2023)

To make more accurate calculations on math tasks. PoT proposed the use of an extended tool to calculate the answer. In their work, they ask the LLM to propose and execute a Python code to get the math answer. In their zero-shot experiment, PoT performs better than zero-shot CoT on GSM8K (57.0%), AQuA (43.9%), SVAMP (70.8%), TabMWP (66.5%), MultiArith (92.2%), and Avg (66.1%).

## 3 HoT

We propose an explainable, logical, and end-to-end hint of thought prompting (HoT) method that uses reasoning paths to represent knowledge to activate more profound reasoning within an LLM. In our work, an end-to-end hints chain (in the step-by-step format) is used as the input of LLMs, and the LLMs itself is required to generate explainable semantic-level step-by-step reasoning steps without any extra prompts. This employs a chain-like instruction to activate more profound knowledge within the LLM. Consequently, the input prompting to the LLM is "[X]. Hints: *hints\_chain*" where [X] is the question, and *hints\_chain* is our HoT prompting.

### 3.1 End-to-end hints chain

The hints chain is a chain of instructions that delivers the LLM with instructions step-by-step. The purpose of the hints chain is to tell the LLM how to generate a reasoning chain. There are three parts of such hints chains: (1) Explainable sub-questions: ask the LLM to partition the question into five step-by-step sub-questions; (2) Logical reasoning: tell the LLM to use the step-by-step sub-questions to

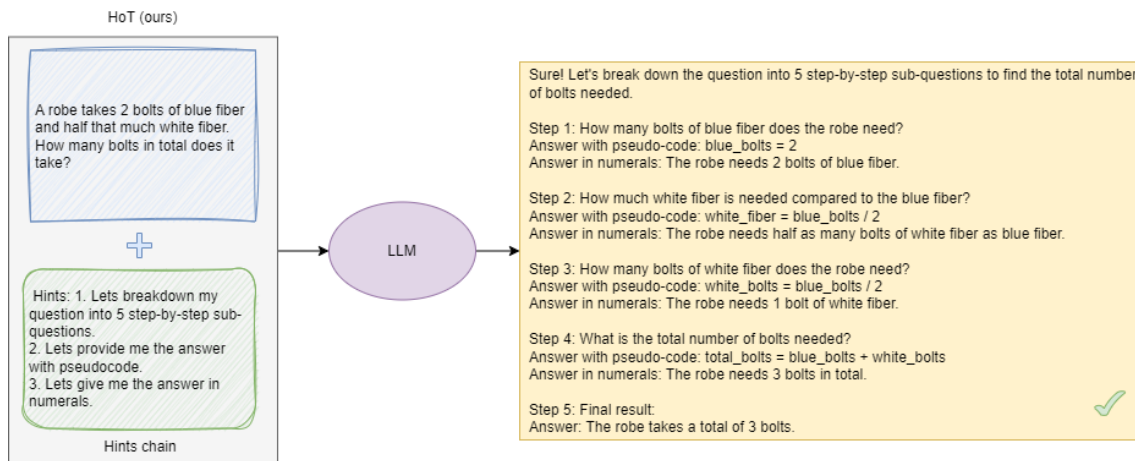


Figure 1: HoT examples on GSM8K

generate a pseudocode and calculate the answer; (3) Answer extraction: ask the LLM to give a customized format of the answer. The result produced by the hints chain follows the order of these three parts and provides a consequential reasoning chain that supports its answer. It is an end-to-end way, meaning we only need to prompt once for a single question to get an answer, which can help the LLM respond quickly.

### 3.2 Explainable question decomposition and sub-question-wise problem solving

Although the prompt in LLM is also in the back box representation, our HoT can produce an understandable and explainable consequent problem-solving procedure. It indeed helps the users to verify the answer more intuitively. Sub-questions not only help the LLM to solve the whole problem but also transparently explain the mind map of the LLM. In our examples (Appendix), the reasoning process and logical reasoning can be clearly seen.

### 3.3 Logical pseudo-code reasoning

An interpretable problem-solving chain can enhance LLM to solve reasoning problems more accurately. However, it is likely to have semantic ambiguity in semantic-level reasoning, although LLM is designed to work with language. Therefore, pseudocode, regarded as a more accurate logical language in programming design, is further exploited.

## 4 Experimental Results

All the experiments done with our HoT are based on the GPT-3.5 family with GPT-3.5-turbo. Our baselines are adopted from zero-shot CoT (Kojima

et al., 2022) with text-davinci-002 as well as PoT (Chen et al., 2023) with code-davinci-002. Additionally, text-davinci-002 and code-davinci-002 also belong to the GPT-3.5 family.

### 4.1 Tasks and datasets

We evaluate the HoT prompting on the five datasets on the four main arithmetic reasoning tasks: GSM8K, AQUA, SVAMP, and ADDSUB. In addition, we complete experiments on the big commonsense reasoning benchmark StrategyQA. Note that all the datasets we utilize in our experiments are publicly released.

Dataset	# of samples
GSM8K	1319
AQUA	254
SVAMP	1000
ADDSUB	395
StrategyQA	2290

Table 1: Number \*# of samples in different datasets

### 4.2 Arithmetic tasks

For the arithmetic reasoning task, the following four benchmarks are considered: i.e., 1) GSM8K (Cobbe et al., 2021), 2) AQUA-RAT (Ling et al., 2017), 3) SVAMP (Patel et al., 2021), and 4) AddSub (Hosseini et al., 2014). The first three datasets, especially GSM8K, are more recently published benchmarks and have more challenges because they require multi-step reasoning to solve problems. Also, the AQUA is a multiple-choice question dataset. The baseline results are quoted from zero-shot CoT (Kojima et al., 2022) and zero-shot experiment done in PoT (Chen et al., 2023).

Method	GSM8K	AQUA
zero-shot CoT	40.50%	31.9%
PoT with zero-shot	57.0%	43.9
HoT (Ours)	<b>67.80%</b>	<b>46.4%</b>

Table 2: Results on GSM8K and AQUA

Method	SVAMP	ADDSUB
zero-shot CoT	63.7%	74.7%
PoT with zero-shot	70.8%	-
HoT (Ours)	<b>76.9%</b>	<b>87.34%</b>

Table 3: Results on SVAMP and ADDSUB

We can observe that HoT has the best performance among the zero-shot reasoning prompts. We design the HoT prompts here, requiring a numeral answer in the third part of the hints chain. From the results, it is easy to see that HoT has a higher potential for arithmetic reasoning than the previous zero-shot approaches.

### 4.3 Commonsense tasks

We employ StrategyQA (Geva et al., 2021) for the commonsense reasoning task, a large dataset that requires the model to conduct implicit multi-hop reasoning to answer questions. The baseline results are from zero-shot CoT. (Kojima et al., 2022). The reasoning accuracy with HoT is about 30% higher than CoT. We make the HoT prompt here, which requires a Yes/No in the third part of the hints chain. It is readily observed from the results that HoT has a higher potential for commonsense reasoning than existing zero-shot approaches.

Method	StrategyQA
zero-shot CoT	52.3%
HoT (Ours)	<b>82.96%</b>

Table 4: Results on StrategyQA

### 4.4 Answer verification

GPT-3.5-turbo always answers (output) in a complete sentence, making it difficult to automatically verify if the output is the same answer as that in the metrics. To check the answer, we independently prompt again to the LLM with its answer as a record. In this verification prompt, we take the model answer as user input and verify if it's the same as the answer given by the LLM.

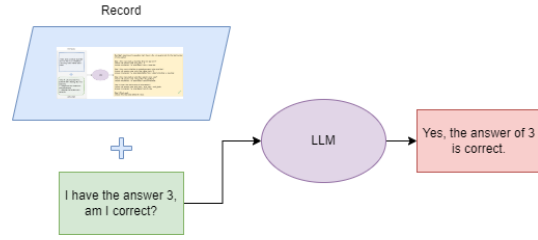


Figure 2: Full pipeline of answer verification

## 5 Error Analysis

We divide the reasoning errors into (1) reasoning errors and (2) calculation errors. Our experimental results illustrate that reasoning errors always occur when the question (Error on GSM8K) has semantic ambiguity or if it is too complex. It is hard to avoid such errors because it is hard even for a human being. Although the calculation error rarely occurs, this can happen, and this can be potentially tackled by adding an external calculator.

## 6 Ablation study

### 6.1 The number of sub-questions.

We conduct HoT experiments with three sub-questions (HoT-3) and seven sub-questions (HoT-7) on the SVAMP dataset, respectively, while the standard one is HoT-5 with five sub-questions. We can see that more sub-questions may not provide a more accurate answer. The LLMs start making stories when those steps are not needed. Also, more sub-questions cost more for the LLMs to respond, which takes more time and effort.

Method	SVAMP
zero-shot CoT	52.3%
PoT with zero-shot	43.9.0%
HoT-3	46.1%
HoT-7	44.4%
HoT-5	46.4%

Table 5: Ablation study on SVAMP

## 7 Related work

### 7.1 Complex reasoning with LLMs

Reasoning skills are essential for general intelligence systems, and the ability to reason in LLMs gained significant attention from the research community. Several studies (Brown et al., 2020; Smith et al., 2022) have shown that asking pre-trained

models to produce step-by-step reasoning or fine-tuning (Cobbe et al., 2021) can increase their ability on complex reasoning tasks. GPT-3 (Brown et al., 2020) has illustrated its robust few-shot reasoning (Wei et al., 2022b; Wang, 2022; Chowdhery et al., 2022) that a few examples in natural language are given to the model to describe the task. The most classic reasoning tasks are mathematical reasoning. PoT (Chen et al., 2023) has shown great ability on math reasoning tasks with LLMs with the help of Python programs. They aim to generate an executive Python program by the LLM to solve math problems. However, their work primarily focuses on math reasoning tasks. A more general approach would be CoT (Wei et al., 2022a), which works well on mathematical, logical, common sense, and symbolic reasoning tasks with few-shot prompts.

## 7.2 Zero-shot reasoning with LLMs

It was indicated that LLMs have excellent zero-shot abilities in many system-1 tasks, including reading comprehension, translation, and summarization (Radford et al., 2019). This ability can also be fine-tuned to get a better performance (Ouyang et al., 2022). However, we focus on system-2 tasks beyond system-1 tasks. The recent work, zero-shot CoT, increases zero-shot performance. Also, PoT in zero-shot format provides good results in math reasoning tasks.

## 7.3 Discussion about Existing Work

Recently, there have been many approaches to enhance the reasoning ability of LLMs, including CoT (Wei et al., 2022a), zero-shot CoT (Kojima et al., 2022), Auto-CoT (Shin et al., 2020), PoT (Chen et al., 2023), decomposed prompting (Khot et al., 2023). They all aim to provide accurate reasoning results. On the other hand, they did not focus on answer explanation and the LLM efficiency. Therefore, HoT is proposed and gives another approach to prompt engineering.

## 8 Discussion

In our work, we approach the reasoning tasks from other points of view. We care about accuracy and want a more precise explanation of the reasoning process. We have verified that HoT can work efficiently on arithmetic and common sense reasoning tasks and provide a clear description intuitively. Although LLMs are black boxes, we tried to explain them with a prompt approach.

## 9 Conclusion

This paper investigates how to generate a more intuitively explainable end-to-end prompting on reasoning tasks. Our experimental results on the mathematical reasoning tasks significantly surpass that of prior zero-shot methods with GPT-3.5 family, CoT, and PoT. Also, on the common sense benchmark StrategyQA, our HoT reaches a new zero-shot GPT-3.5 result with 82.96%.

## 10 Limitation and Social Impact

Our work is based on GPT-3, a pre-trained language model trained from various sources and shown to capture and amplify biases found in the training data. We use prompting to guarantee our reasoning answers, which takes advantage of the patterns learned by language models. However, our zero-shot approach directly probes complex reasoning inside pre-trained LLM, which can also cause bias. Also, our ability to reason is based on the power of the LLM. Therefore, the accuracy depends on GPT-3, which may cause fluctuation in different test environments.

## References

- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, et al. 2020. [Language models are few-shot learners.](#)
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. [Sparks of artificial general intelligence: Early experiments with gpt-4.](#)
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. [Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.](#)
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, et al. 2022. [Palm: Scaling language modeling with pathways.](#)
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems.](#)
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of](#)

394	deep bidirectional transformers for language understanding.	OpenAI. 2023. Gpt-4 technical report.	450
395	In <i>Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)</i> , pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.		
396		Siqi Ouyang, Rong Ye, and Lei Li. 2022. On the impact of noises in crowd-sourced data for speech translation.	451
397		In <i>Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)</i> , pages 92–97, Dublin, Ireland (in-person and online). Association for Computational Linguistics.	452
398			453
399			454
400			455
401	Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling.	Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems?	456
402		In <i>Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies</i> , pages 2080–2094, Online. Association for Computational Linguistics.	457
403			458
404			459
405			460
406	Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners.	Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.	461
407	In <i>Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)</i> , pages 3816–3830, Online. Association for Computational Linguistics.		462
408			463
409			464
410			465
411			466
412			467
413			468
414	Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies.	Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, et al. 2022. Scaling language models: Methods, analysis insights from training gopher.	469
415	In <i>Transactions of the Association for Computational Linguistics</i> , 9:346–361.		470
416			471
417			472
418			473
419			474
420	Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization.	Timo Schick and Hinrich Schütze. 2021. It’s not just size that matters: Small language models are also few-shot learners.	475
421	In <i>Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)</i> , pages 523–533, Doha, Qatar. Association for Computational Linguistics.	In <i>Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies</i> , pages 2339–2352, Online. Association for Computational Linguistics.	476
422			477
423			478
424			479
425			480
426			481
427	Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks.	Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts.	482
428		In <i>Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)</i> , pages 4222–4235, Online. Association for Computational Linguistics.	483
429			484
430			485
431	Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners.	Eric Smith, Orion Hsu, Rebecca Qian, Stephen Roller, Y-Lan Boureau, and Jason Weston. 2022. Human evaluation of conversations is an open problem: comparing the sensitivity of various methods for evaluating dialogue agents.	486
432	In <i>Advances in Neural Information Processing Systems</i> , volume 35, pages 22199–22213. Curran Associates, Inc.	In <i>Proceedings of the 4th Workshop on NLP for Conversational AI</i> , pages 77–97, Dublin, Ireland. Association for Computational Linguistics.	487
433			488
434			489
435			490
436	Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems.	Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, et al. 2022. Lamda: Language models for dialog applications.	491
437	In <i>Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 158–167, Vancouver, Canada. Association for Computational Linguistics.		492
438			493
439			494
440			495
441			496
442			497
443	Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing.	Zhengxiang Wang. 2022. Linguistic knowledge in data augmentation for natural language processing: An example on Chinese question matching.	498
444		In <i>Proceedings of the 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022)</i> , pages	499
445			500
446			501
447	Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.		502
448			503
449			504
			505

506	40–49, Trento, Italy. Association for Computational Linguistics.	Therefore, Eliza’s earnings for this week are \$460.	558
507			559
508	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022a. <a href="#">Chain-of-thought prompting elicits reasoning in large language models.</a>		560
509		<b>A.2 Example on AQUA</b>	561
510		Question: Find out which of the following values is the multiple of X, if it is divisible by 9 and 12? [’A)36’, ’B)15’, ’C)17’, ’D)5’, ’E)7’]	562
511		Answer: A	563
512	Kangda Wei, Sayan Ghosh, and Shashank Srivastava. 2022b. <a href="#">Compositional generalization for kinship prediction through data augmentation.</a> In <i>Proceedings of the 4th Workshop of Narrative Understanding (WNU2022)</i> , pages 13–19, Seattle, United States. Association for Computational Linguistics.	Output:	564
513		Step 1: Check if the given values are divisible by both 9 and 12.	565
514			566
515			567
516			568
517			569
518	Junjie Ye, Xuantang Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. <a href="#">A comprehensive capability analysis of gpt-3 and gpt-3.5 series models.</a>	- 36 is divisible by both 9 and 12.	570
519		- 15 is not divisible by 9.	571
520		- 17 is not divisible by 9 or 12.	572
521		- 5 is not divisible by 9.	573
522		- 7 is not divisible by 9 or 12.	574
523			575
524	<b>A Appendix</b>	Step 2: Identify which value is divisible by both 9 and 12.	576
525	<b>A.1 Example on GSM8K</b>		577
526	Question: Eliza’s rate per hour for the first 40 hours she works each week is \$10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week?	- The multiple of both 9 and 12 is 36.	578
527	Answer: 460	Therefore, the answer is A) 36.	579
528	Output:		580
529	Sure! Let’s break down the question into step-by-step sub-questions and provide answers in pseudocode and numerals:		581
530			582
531	1. Calculate Eliza’s regular earnings for the first 40 hours at a rate of \$10 per hour:	<b>A.3 Example on SVAMP</b>	583
532	a. Pseudo-code: <code>regular_earnings = rate_per_hour * 40</code>	Question: Each pack of dvds costs 76 dollars. If there is a discount of 25 dollars on each packHow much do you have to pay to buy each pack?	584
533	b. Numerical calculation: <code>regular_earnings = \$10/hour * 40 hours = \$400</code>	Answer: 51.0	585
534		Output:	586
535		Sure! Let’s break down the question into five step-by-step sub-questions:	587
536			588
537		1. What is the original cost of each pack of DVDs?	589
538		Answer: \$76	590
539			591
540		2. What is the discount on each pack?	592
541		Answer: \$25	593
542			594
543		3. What is the new cost of each pack after the discount?	595
544		Answer: Original cost - Discount = \$76 - \$25 = \$51	596
545			597
546		4. How much do you have to pay to buy each pack?	598
547		Answer: \$51	599
548			600
549			601
550			602
551			603
552			604
553			605
554			606
555			
556			
557			

607	Now, let me provide you with the answers in	step-by-step sub-questions and provide answers	657
608	pseudocode :	using pseudocode :	658
609			659
610	original_cost = 76	1. Is September 11th a memorial day?	660
611	discount = 25	- Answer: Yes	661
612	new_cost = original_cost - discount		662
613		2. Does the September 11th memorial commem-	663
614	And finally, in numerals, the answer is: \$51.	orate the tragic events of the terrorist attacks on	664
615		September 11, 2001?	665
616		- Answer: Yes	666
617	<b>A.4 Example on ADDSUB</b>		667
618	Question: Joan found 70 seashells on the beach .	3. Were firefighters involved in responding to	668
619	she gave Sam some of her seashells . She has 27	the terrorist attacks on September 11, 2001?	669
620	seashell . How many seashells did she give to Sam	- Answer: Yes	670
621	?		671
622	Answer: 43	4. Are individuals who were involved in re-	672
623	Output:	sponding to the terrorist attacks on September 11th,	673
624	Sure! Let's break down the question into step-by-	including firefighters, honored and remembered in	674
625	step sub-questions and provide the answers using	the September 11th memorial?	675
626	pseudocode first:	- Answer: Yes	676
627			677
628	1. How many seashells did Joan originally	5. Are firefighters included as part of the	678
629	have?	memorial tribute for their bravery and sacrifice	679
630	Answer: Joan originally had 70 seashells.	during the events of September 11th?	680
631		- Answer: Yes	681
632	2. How many seashells does Joan have after		682
633	giving some to Sam?	So, in summary, the answer to your question is:	683
634	Answer: Joan now has 27 seashells.	Yes, firefighters are included in the September 11th	684
635		memorial.	685
636	3. How many seashells did Joan give to Sam?		686
637	Answer: To find the number of seashells Joan gave	Please note that the pseudocode used here is	687
638	to Sam, we can subtract the number of seashells	to illustrate the logical flow of the sub-questions	688
639	she has now (27) from the number she originally	and their answers. The actual implementation may	689
640	had (70).	vary based on the specific context or programming	690
641	Pseudocode: Joan's seashells given to Sam = 70 -	language.	691
642	27		692
643	4. Convert the pseudocode into numerals:	<b>A.6 Error example on GSM8k</b>	693
644	Answer: Joan gave Sam 43 seashells.	Question: An oil pipe in the sea broke . Before	694
645		engineers started to fix the pipe , 2475 gallons of	695
646	So, Joan gave Sam 43 seashells.	oil leaked into the water . A total of 6206 gallons	696
647		of oil leaked before the pipe was fixed . How many	697
648	<b>A.5 Example on StrategyQA</b>	gallons of oil leaked while the engineers were	698
649	Question: Would Firefighters be included in a	fixing the pipe ?	699
650	September 11th memorial?	Answer: 3731.0	700
651	Answer: Yes. September 11th is remembered as a	Output:	701
652	day of mourning for the lives lost during a terrorist	Sure! Let's break down the problem into step-by-	702
653	attack in NYC. Firefighters were among the first	step sub-questions:	703
654	responders to the crisis, and many died.		704
655	Output:	Step 1: Calculate the total amount of oil leaked	705
656	Sure! Let's break down your question into five	after the engineers fixed the pipe.	706
		This can be found by subtracting the amount	707



708 leaked before the engineers started from the total  
709 amount leaked.

710 Total leaked after fixing = Total leaked - Leaked  
711 before fixing

712 Total leaked after fixing = 6206 gallons - 2475  
713 gallons

715 Step 2: Calculate the amount of oil leaked while  
716 the engineers were fixing the pipe.

717 This can be found by subtracting the total leaked  
718 after fixing from the amount leaked before fixing.

719 Leaked while fixing = Leaked before fixing - Total  
720 leaked after fixing

722 Step 3: Convert the answer from step 2 into  
723 numerals.

725 Now let's provide you with the pseudocode :

727 Step 1:

728 Define the variable "totalLeaked" and assign it the  
729 value of 6206 gallons.

730 Define the variable "leakedBeforeFixing" and  
731 assign it the value of 2475 gallons.

732 Define the variable "totalLeakedAfterFixing" and  
733 assign it the result of subtracting leakedBeforeFix-  
734 ing from totalLeaked.

736 Step 2:

737 Define the variable "leakedWhileFixing" and  
738 assign it the result of subtracting totalLeakedAfter-  
739 Fixing from leakedBeforeFixing.

741 Step 3:

742 Print the value of leakedWhileFixing.

744 The answer in numerals is the value of leaked-  
745 WhileFixing obtained from the pseudocode.

746 No, your answer is not correct. Let's go through  
747 the calculations again:

749 Step 1: Total leaked after fixing = 6206 gallons -  
750 2475 gallons = 3731 gallons

752 Step 2: Leaked while fixing = 2475 gallons -  
753 3731 gallons = -1256 gallons

755 It seems that we made a mistake in the calcula-  
756 tion. The result is a negative value, which doesn't  
757 make sense in this context. Let's double-check the  
758 steps:

760 Step 1: Total leaked after fixing = 6206 gallons -  
761 2475 gallons = 3731 gallons

763 Step 2: Leaked while fixing = 3731 gallons -  
764 2475 gallons = 1256 gallons

766 So, the correct answer is 1256 gallons.  
767