
Appendix (Supplementary Material)

Anonymous Author(s)

Affiliation

Address

email

A Limitations

We are encouraged by the promising improvements observed on general reasoning datasets through the application of RL with our 92K sample data. However, for 7B and 32B models, our current training only lasts two epochs due to computing resource constraints. It is conceivable that extended training might yield even more pronounced results.

Our approach initiated training directly from a base model using RL, bypassing an initial SFT phase. This decision was made to ensure that our conclusions regarding cross-domain generalization remained independent of specific SFT data dependencies. However, we expect that integrating an SFT then RL pipeline could further enhance performance largely, and we plan to explore this in future work.

B Data

In this section, we elaborate the data sourcing and curation details of all used dataset in GURU.

New Assets: We're committed to full transparency and open science. All the new assets we've created, as well as those repurposed from existing datasets, are fully open-source and publicly available. We believe this approach fosters collaboration and accelerates progress within the research community.

B.1 Math

For our math subset, we focus on gathering some of the most effective datasets recently released. We prioritize OR1 [He et al., 2025], DAPO [Yu et al., 2025], and DeepScaler [Luo et al., 2025b] due to their strong performance in advancing mathematical reasoning in large language models. For other math reasoning datasets, such as BigMath [Albalak et al., 2025], we observe that their quality is inconsistent, and they contain more duplicate queries, which led us to exclude them from our selection. We identify the original sources of these datasets and, as shown in Figure 1, meticulously trace their origins to understand their provenance and potential overlaps. We further standardize the data formats across all selected datasets, ensuring consistency and ease of integration into our training pipeline. A critical step is identifying and eliminating overlapping or duplicate problems within and across datasets. This ensures that our model is exposed to a diverse set of unique challenges, preventing overfitting to specific problem types and promoting broader generalization. By carefully curating these datasets, we aim to provide a robust and varied foundation for training and evaluating advanced mathematical reasoning capabilities in LLMs.

B.2 Code

Our code subset consists of programming problems collected from a variety of sources, including online coding platforms and programming competitions. Specifically, we include real-world challenges from LeetCode [Xia et al., 2025], curated and validated tasks from TACO-Verified [Li, 2024], problems from PrimeIntellect [Mattern et al., 2025], and historical contest problems from

Table 1: Public source data links for all datasets used in each domain for GURU.

Domain	Dataset	Data Source Link
Math	OR1	🤗 Skywork/Skywork-OR1-RL-Data
	DAPO	🤗 BytedTsinghua-SIA/DAPO-Math-17k
	DeepScaler	🤗 agentica-org/DeepScaleR-Preview-Dataset
Code	LeetCode	🤗 newfacade/LeetCodeDataset
	TACO-Verified	🤗 likaixin/TACO-verified
	PrimeIntellect	🤗 agentica-org/DeepCoder-Preview-Dataset
	LiveCodeBench	🤗 agentica-org/DeepCoder-Preview-Dataset
Science	WebInstruct-Verified	🤗 TIGER-Lab/WebInstruct-verified
Logic	Zebra Puzzle	–
	Ordering Puzzle	–
	Graph Puzzle	–
	ARC-AGI	🤗 fchollet/ARC-AGI
	ARC-AGI-2	🤗 arcprize/ARC-AGI-2
	BARC	🤗 barc0/200k_HEAVY_gpt4o...problems
Simulation	Code I/O (PyEdu)	🤗 hkust-nlp/CodeIO-PyEdu-Reasoning-Raw
Tabular	HiTab	🤗 microsoft/HiTab
	MultiHierTT	🤗 psunlpgroup/MultiHiertt

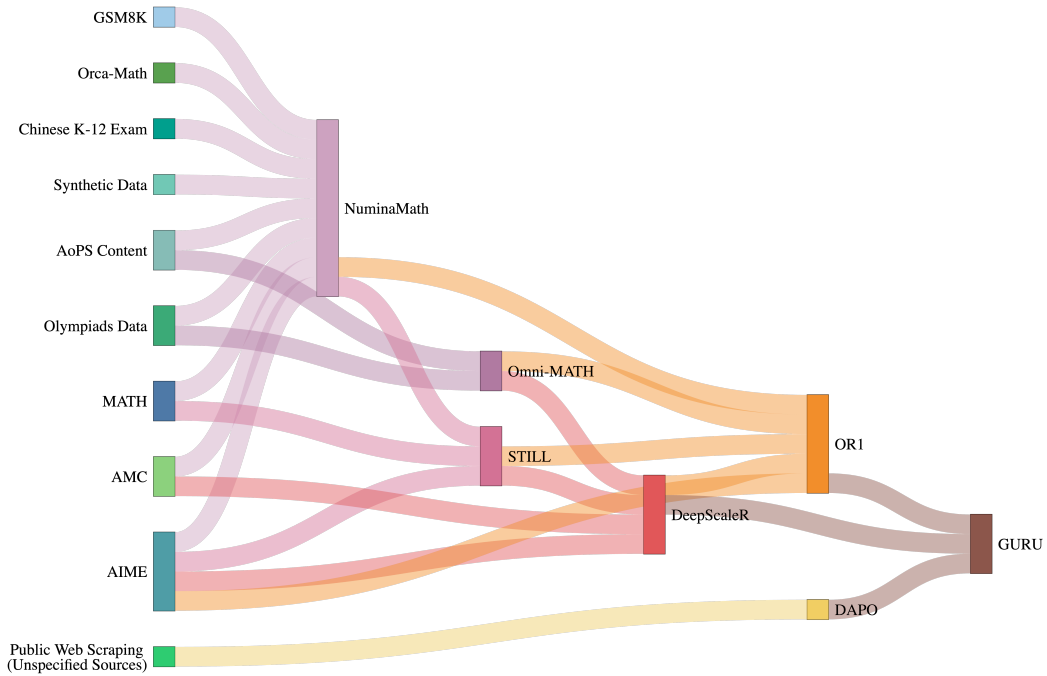


Figure 1: The sankey diagram tracing the provenance of all math data included in GURU.

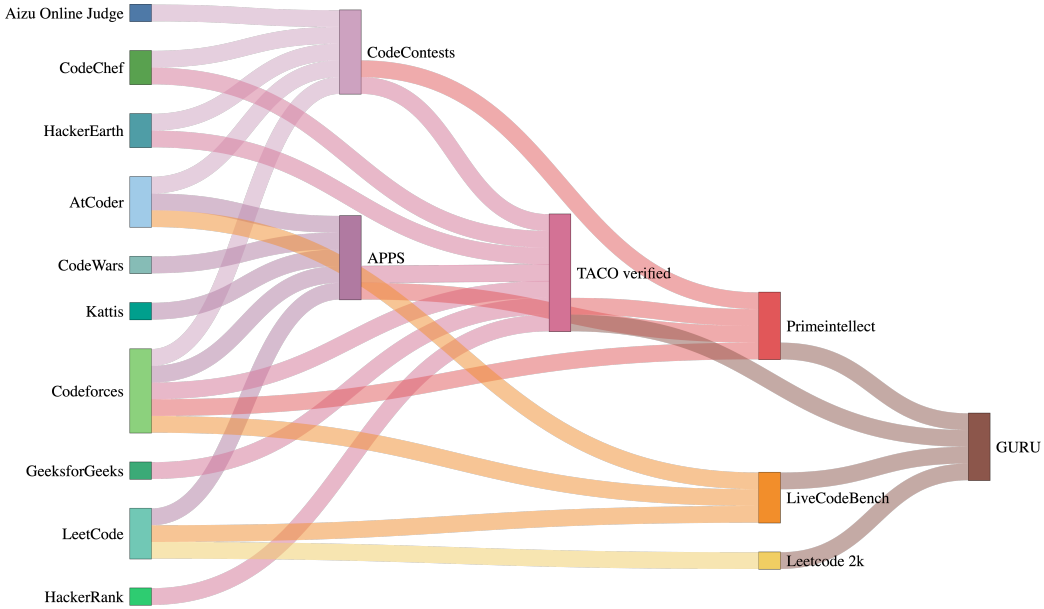


Figure 2: The sankey diagram tracing the provenance of all code data included in GURU.

LiveCodeBench [Jain et al., 2024]. For PrimeIntellect and LiveCodeBench, we adopt the pre-filtered subsets provided by the DeepCoder dataset [Luo et al., 2025a], which applies rigorous cleaning and verification protocols. A comprehensive breakdown of data source and corresponding Hugging Face links is provided in Table 1.

Each problem is presented as a program synthesis task requiring the model to generate a Python solution that satisfies a given problem specification to pass certain test cases. Specifically, we have two problem formats: (1) *pure function format*, where a function is implemented and validated via inline `assert` statements; and (2) *stdin/stdout format*, where the model must write a full script that reads from standard input and writes to standard output. All program executions are sandboxed with a uniform 30-second timeout to ensure stability and computational efficiency and prevent infinite loops. Our current setup only focuses on Python, we leave the inclusion of additional programming languages (e.g., Java, C++, Go, etc.) as future work, to explore both cross-lingual transfer and the necessity of language-level diversity in RL training for code.

To provide full transparency on how different datasets relate and evolve, we construct a Sankey diagram (Figure 2) that traces the provenance of all code data included in our benchmark.

B.3 Science

Our science data leverages WebInstruct-Verified [Ma et al., 2025]. This dataset comprises over 230K verifiable question-answer pairs across diverse scientific disciplines, including physics, chemistry, biology, and finance. To ensure the dataset’s suitability for reinforcement learning, we applied a rigorous filtering process, detailed in the main text, to select examples that provide reliable supervision signals.

B.4 Logic

To evaluate and improve the logical reasoning capabilities of language models, we design three synthetic tasks and collect three existing datasets, all of which require multi-step deduction under structured constraints.

Zebra Puzzles Zebra puzzles are a class of logic grid problems that require deducing a complete set of attribute assignments (e.g., who owns the zebra) based on a set of relational constraints. The task becomes exponentially more difficult as the number of entities and constraints increases, making it a strong benchmark for multi-step reasoning. We construct puzzles by first sampling a ground-truth table over randomly chosen attribute categories and values, then back-solving to generate constraints. These include positional relations (e.g., left/right, middle), equality/inequality constraints, disjunctive logic, relative distances, and parity-based rules. We define 20 difficulty levels based on constraint complexity, support puzzles up to 15×10 in size, and optionally include redundant constraints to increase distractor robustness.

Ordering Puzzles Ordering puzzles require models to recover a linear order over n objects (up to 50) based on a set of relational constraints. We first sample a ground-truth permutation and then generate constraints from it. These include absolute positions (e.g., “A is at position 3”), relative positions (e.g., “A is to the left of B”), adjacency relations (e.g., “A is immediately to the right of B”), and distance-based constraints (e.g., “There are 3 items between A and B”). To ensure uniqueness, we apply constraint propagation to select a minimal satisfiable subset. Our dataset spans diverse entity types, such as animals, people, and vehicles.

Graph Puzzles Graph puzzles task models with identifying a valid path through a directed graph defined by natural language implications. We build upon the structure of PrOntoQA [Saparov and He, 2023], extending it to free-form language using the setup of Graph Search Tasks [Saparov et al., 2025]. Each instance defines a set of predicates that implicitly specify the graph structure, along with a query that requires multi-hop traversal. We vary the number of nodes, edge density, and path length to control task difficulty, with longer and denser paths posing greater challenges for compositional reasoning.

ARC-AGI The original *Abstraction and Reasoning Corpus for Artificial General Intelligence* [Chollet et al., 2024] (ARC-AGI-1) comprises 400 public training tasks and an equally sized public evaluation split, all expressed as grid-based input–output transformations. Tasks are deliberately designed to be straightforward for humans yet challenging for machines, probing fluid, compositional intelligence over a space of 30×30 coloured grids. Each instance supplies 2–5 demonstration pairs (as input examples) and one unseen test grid. We include the full training split of ARC-AGI-1 when constructing GURU.

ARC-AGI-2 ARC-AGI-2 [Chollet et al., 2025] expands to ARC-AGI to 1,000 public training tasks and 120 public evaluation tasks while introducing two hidden private test sets for competition use. The new version re-balances difficulty so that average human solvers achieve roughly 66% accuracy, sharpens the trial budget to two attempts per test grid, and retains the canonical grid-world format. By fusing novel tasks with the ARC-AGI-1 corpus, ARC-AGI-2 covers a broader concept space (symmetry, physics-like dynamics, object counting, etc.) while preserving the “few demonstrations, zero prior training” ethos. We include the public training portion in GURU.

BARC BARC [Li et al., 2024] is a large-scale synthetic extension containing approximately 200,000 ARC-style problems generated from 160 manually curated seed programs via an LLM-guided remix pipeline. Each synthetic task is accompanied by executable Python reference code and an input generator, enabling controllable difficulty and unlimited fresh samples for augmentation or test-time fine-tuning. We sample $3.4k$ examples whose input lengths do not exceed the token limit of our training.

103 B.5 Simulation

Simulation is the process of creating a model of a real or hypothetical system and executing it to observe its dynamic behavior over time. This process is inherently linked to reasoning because it requires the systematic application of logical inference to predict the consequences of actions or changes within a system. In this domain, our work focuses on the specific subproblem of *simulating Python program execution*. Prior research has explored various facets of this problem. [Zaremba and Sutskever, 2014] trained LSTM to evaluate short computer programs. [Nye et al., 2021] demonstrated the capability of LLMs to predict code execution traces. CruxEval [Gu et al., 2024] further advanced this by introducing a benchmark to evaluate LLM performance in both inferring inputs from given

function outputs and predicting outputs from given inputs. Recent works, such as SemCoder [Ding et al., 2024] and Code I/O [Li et al., 2025], have used SFT with code execution prediction tasks to enhance LLM capabilities in code generation and general reasoning.

In GURU, we repurpose the raw data from Code I/O for our RL training. The Code I/O dataset, sourced from PyEdu [HuggingFaceTB and PyEdu Developers, 2024], comprises 4.5 million verified programs with corresponding input-to-output and output-to-input samples. Inputs for this dataset were generated by DeepSeek-v2.5, and outputs were obtained through actual function execution. To ensure diversity within our training data, we first applied a heuristic filter to identify unique programs, reducing the dataset to 227K programs. Subsequently, we randomly sampled 12K I/O pairs. A final difficulty-based filtering step was then applied, resulting in a refined dataset of 3.7K I/O samples used for our RL training.

B.6 Tabular

Tabular data are ubiquitous and serve as vital media for data organization and analysis. Table Question Answering (TQA), which involves deriving an answer from one or more tables given a natural language query, is a fundamental and critical problem in data understanding. Early work in TQA, such as WikiTQ [Pasupat and Liang, 2015] and HybridQA [Chen et al., 2020], explored answering queries on WikiTables, sometimes incorporating accompanying text. The complexity of TQA increased with datasets like HiTab [Cheng et al., 2021] and MultiHiertt [Zhao et al., 2022], which focused on hierarchical tables featuring multiple levels of headers, demanding more sophisticated reasoning. More specialized applications emerged with FinQA [Chen et al., 2021] and ConvFinQA [Chen et al., 2022], addressing TQA within the financial domain.

In GURU, we adapt the training sets from HiTab (7.5K samples) and MultiHiertt (7.8K samples) for our RL training. We chose these datasets because they necessitate complex multi-hop reasoning over tabular structures. For table representation, we linearize the tables into a markdown format. If multiple tables are present for a single query, we concatenate them, separated by a line break.

C Examples

Math

Query:

For complex number $u = a + bi$ and $v = c + di$ (where $i = \sqrt{-1}$), define the binary operation

$$u \otimes v = ac + bdi$$

Suppose z is a complex number such that $z \otimes z = z^2 + 40$. What is $|z|^2$? Please output the final answer within `\boxed{ }`.

Response:

<think>

Let's break down the problem step-by-step:

1. We are given the binary operation $u \otimes v = ac + bdi$ for complex numbers $u = a + bi$ and $v = c + di$.
2. We need to find a complex number z such that $z \otimes z = z^2 + 40$.
3. Let $z = x + yi$, where x and y are real numbers.
4. First, compute $z \otimes z$:

$$z \otimes z = (x + yi) \otimes (x + yi) = x \cdot x + y \cdot yi = x^2 + y^2i$$

5. Next, compute z^2 :

$$z^2 = (x + yi)^2 = x^2 + 2xyi + (yi)^2 = x^2 + 2xyi - y^2 = (x^2 - y^2) + 2xyi$$

6. We are given that $z \otimes z = z^2 + 40$:

$$x^2 + y^2i = (x^2 - y^2) + 2xyi + 40$$

7. Equate the real and imaginary parts:

$$x^2 = x^2 - y^2 + 40 \quad (\text{real part})$$

$$y^2 i = 2xyi \quad (\text{imaginary part})$$

8. From the imaginary part equation:

$$y^2 = 2xy$$

If $y \neq 0$, we can divide both sides by y :

$$y = 2x$$

9. From the real part equation:

$$x^2 = x^2 - y^2 + 40$$

Simplify:

$$0 = -y^2 + 40$$

$$y^2 = 40$$

10. If $y = 2x$, substitute $y^2 = 40$ into $y = 2x$:

$$(2x)^2 = 40$$

$$4x^2 = 40$$

$$x^2 = 10$$

11. Now, we need to find $|z|^2$:

$$|z|^2 = x^2 + y^2$$

Substitute $x^2 = 10$ and $y^2 = 40$:

$$|z|^2 = 10 + 40 = 50$$

</think>

50

139

Code

Query:

Write a python function to find the frequency of the largest value in a given array.

Your solution should be a complete, self-contained function in a markdown code block. Make sure your solution passes the following test cases:

```
assert frequency_Of_Largest(5,[1,2,3,4,4]) == 2
assert frequency_Of_Largest(3,[5,6,5]) == 1
assert frequency_Of_Largest(4,[2,7,7,7]) == 3
```

Please do not include the test cases in your solution.

Response:

<think>

Let's break down the problem and figure out how to solve it:

1. We need to find the frequency of the largest value in a given array.
2. First, we need to identify the largest value in the array.
3. Then, we need to count how many times this largest value appears in the array.
4. We can achieve this by:
 - Finding the maximum value in the array.

140

- Counting the occurrences of this maximum value in the array.

5. We will write a function that takes the size of the array and the array itself as input and returns the frequency of the largest value.

</think>

Here is the complete, self-contained function to solve the problem:

```
def frequency_Of_Largest(n, arr):
    # Find the maximum value in the array
    max_value = max(arr)
    # Count the occurrences of the maximum value in the array
    frequency = arr.count(max_value)
    return frequency
```

This function first finds the maximum value in the array using the `max()` function. Then, it counts the occurrences of this maximum value using the `count()` method of the list. The function returns the frequency of the largest value in the array.

141

Science

Query:

What is the correct answer to this question:

Astronomers are studying a system of five exoplanets with circular orbits, discovered through the TTV method. The planets (designated as Planet_1 through Planet_5), which are in orbital resonance, have orbital periods in a ratio of 1:2:2.5:3.5:5. Assuming these planets possess the same albedo, what is the ratio of the equilibrium temperatures between Planet_4 and Planet_2?

Choices:

- (A) ~ 0.57
- (B) ~ 0.75
- (C) ~ 0.69
- (D) ~ 0.83

Reason through your answer step-by-step. Then, based on your reasoning, provide the single most likely answer choice. Answer in the format "The correct answer is (insert answer here)."

Response:

<think>

Let's break down the problem step-by-step:

1. We are given the orbital periods of five exoplanets in a ratio of 1:2:2.5:3.5:5.
2. We need to find the ratio of the equilibrium temperatures between Planet_4 and Planet_2, assuming they have the same albedo.
3. The equilibrium temperature T of a planet is given by the formula:

$$T = T_{\text{eq}} = T_{\star} \left(\frac{R_{\star}}{2a} \right)^{1/2} \left(\frac{1 - A}{4} \right)^{1/4}$$

where T_{\star} is the temperature of the star, R_{\star} is the radius of the star, a is the semi-major axis of the planet's orbit, and A is the albedo of the planet. 4. Since the albedo A is the same for both planets, it will not affect the ratio of their equilibrium temperatures. 5. The relationship between the orbital period P and the semi-major axis a is given by Kepler's third law:

$$P^2 \propto a^3$$

142

or

$$\frac{P_1^2}{P_2^2} = \frac{a_1^3}{a_2^3}$$

6. Let's denote the orbital period of Planet_1 as P_1 . Then the orbital periods of the other planets are:

$$P_2 = 2P_1, \quad P_3 = 2.5P_1, \quad P_4 = 3.5P_1, \quad P_5 = 5P_1$$

7. Using Kepler's third law, we can find the ratio of the semi-major axes:

$$\left(\frac{P_4}{P_2}\right)^2 = \left(\frac{a_4}{a_2}\right)^3$$

Substituting the given orbital periods:

$$\left(\frac{3.5P_1}{2P_1}\right)^2 = \left(\frac{a_4}{a_2}\right)^3$$

Simplifying:

$$\left(\frac{3.5}{2}\right)^2 = \left(\frac{a_4}{a_2}\right)^3$$

$$\left(\frac{7}{4}\right)^2 = \left(\frac{a_4}{a_2}\right)^3$$

$$\left(\frac{49}{16}\right) = \left(\frac{a_4}{a_2}\right)^3$$

Taking the cube root of both sides:

$$\frac{a_4}{a_2} = \left(\frac{49}{16}\right)^{1/3}$$

8. The equilibrium temperature is inversely proportional to the square root of the semi-major axis:

$$\frac{T_4}{T_2} = \sqrt{\frac{a_2}{a_4}}$$

Substituting the ratio of the semi-major axes:

$$\frac{T_4}{T_2} = \sqrt{\frac{1}{\left(\frac{49}{16}\right)^{1/3}}}$$

Simplifying:

$$\frac{T_4}{T_2} = \left(\frac{16}{49}\right)^{1/6}$$

$$\frac{T_4}{T_2} = \left(\frac{2^4}{7^2}\right)^{1/6}$$

$$\frac{T_4}{T_2} = \left(\frac{2^{4/6}}{7^{2/6}}\right)$$

$$\frac{T_4}{T_2} = \left(\frac{2^{2/3}}{7^{1/3}}\right)$$

$$\frac{T_4}{T_2} = \left(\frac{2^2}{7}\right)^{1/3}$$

$$\frac{T_4}{T_2} = \left(\frac{4}{7}\right)^{1/3}$$

9. Calculating the value:

$$\left(\frac{4}{7}\right)^{1/3} \approx 0.83$$

</think>

The correct answer is (D).

Query:

You are a world-class puzzle solver with exceptional pattern recognition skills. Your task is to analyze puzzles, spot patterns, and provide direct solutions. Given input-output grid pairs as reference examples, carefully observe the patterns to predict the output grid for new test input. Each pair follows the same transformation rule. Grids are 2D arrays. Here are the input and output grids for the reference examples:

Example 1

Input: [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 5, 0, 0, 0, 0, 0, 0, 0], [0, 0, 5, 0, 0, 0, 0, 4, 0, 0], [0, 0, 5, 0, 0, 0, 4, 0, 2, 0], [0, 0, 5, 0, 0, 0, 4, 0, 2, 0], [0, 0, 5, 0, 0, 0, 0, 0, 2, 0], [0, 0, 0, 5, 0, 0, 0, 0, 0, 0], [4, 0, 0, 0, 5, 0, 0, 0, 0, 0], [4, 0, 0, 5, 0, 0, 0, 0, 0, 0], [4, 0, 0, 0, 5, 0, 0, 0, 0, 0]]

Output: [[2], [2], [2], [2]]

Example 2

Input: [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 8, 2, 0, 0, 0, 6, 6, 0], [0, 0, 2, 8, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [8, 2, 0, 0], [6, 6, 6, 0, 0, 0, 2, 8, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 7, 7, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

Output: [[7, 7], [7, 7]]

Example 3

Input: [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 6, 6, 6, 0, 0, 0, 0, 0], [0, 0, 0, 6, 0, 0, 0, 0, 2, 2, 2, 0, 0], [0, 0, 0, 0, 0, 0, 0, 2, 3, 2, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 0], [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0], [0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0], [0, 2, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

Output: [[2, 1, 2], [1, 1, 1]]

Now, solve the following puzzle based on its input grid by applying the rules you have learned from the training data.

[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 4, 8, 0, 0, 0], [0, 0, 0, 0, 0, 0, 8, 8, 0, 0, 0, 0], [0, 4, 4, 0, 0, 0, 8, 4, 0, 0, 0, 0], [0, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 5], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

What is the output grid? Please put your answer within <answer> and </answer> tags, your final answer should be only the output grid (2d array).

Response:

Let's break down the problem step-by-step:

1. We have several input-output pairs, and we need to identify the pattern or transformation rule applied to each input to produce the corresponding output.
2. In Example 1, the input grid has a specific pattern of numbers, and the output grid is a simpler version of the input grid, focusing on a particular section.
3. In Example 2, the input grid has a different pattern, but the output grid still seems to focus on a specific section of the input grid.
4. In Example 3, the input grid has yet another pattern, and the output grid again focuses on a

specific section of the input grid.

5. Let's look at the input grid for the puzzle we need to solve:

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 4, 8, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 8, 8, 0, 0, 0, 0],
 [0, 4, 4, 0, 0, 0, 0, 8, 4, 0, 0, 0, 0],
 [0, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 5, 5, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

6. If we look at the patterns in the previous examples, we can see that the output grids focus on specific sections of the input grids, usually the non-zero elements in a particular area.

7. In this input grid, we can see that the non-zero elements in the middle section (columns 7 to 9) form a pattern that could be the focus of the output grid.

8. The pattern in columns 7 to 9 is:

```
[0, 0, 0],
 [0, 0, 0],
 [4, 8, 0],
 [8, 8, 0],
 [8, 4, 0],
 [0, 0, 0],
 [0, 0, 0],
 [0, 0, 0],
 [0, 0, 0],
 [0, 0, 0],
 [0, 0, 0],
 [0, 0, 0],
 [0, 0, 0],
 [0, 0, 0]]
```

9. If we extract this pattern and simplify it, we get:

```
[[4, 8],
 [8, 8],
 [8, 4]]
```

</think>

<answer>[[4, 8], [8, 8], [8, 4]]</answer>

145

Simulation

Query:

You are given a question that requires some input and output variables as follows: Given the birthday of a person and the current date, how many days have passed since their birthday? The dates are provided in the format of year, month, and day. Ensure that the dates are valid and account for leap years in your calculation. The input and output requirements are as follows:

Input:

- year1 (int): The year of the first date.
- month1 (int): The month of the first date (1-12).
- day1 (int): The day of the first date (1-31).
- year2 (int): The year of the second date.
- month2 (int): The month of the second date (1-12).
- day2 (int): The day of the second date (1-31).

Output:

146

- **return** (int or str): The number of days between the two dates. If the input is invalid, it returns 'Input Error'.

Given the following output:

6550

Can you predict a feasible input without writing any code? Please reason and put your final answer in the following json format: "input": , where should be a dictionary, even if there is only one input variable, with keys strictly match the input variables' names as specified.

Tip: Here is a reference code snippet for this question. You can refer to this code to guide your reasoning but not copy spans of code directly.

```
# import necessary packages
import random

# all class and function definitions in the code file , if any
def leap_year(year):
    return year % 4 == 0 and (year % 100 != 0 or year % 400
    == 0)

def day_in_month(month, year):
    if month in [4, 6, 9, 11]:
        return 30
    elif month == 2:
        return 29 if leap_year(year) else 28
    else:
        return 31

def day_in_year(year):
    return 366 if leap_year(year) else 365

def dayBetweenMonths(month1, year1, month2, year2):
    b, c = 0, 0
    while month1 < 12:
        b += day_in_month(month1, year1)
        month1 += 1
    while month2 < 12:
        c += day_in_month(month2, year2)
        month2 += 1
    return (b - c) if year1 == year2 else (b - day_in_year(
    year1) + day_in_year(year2) - c)

# main function
def main_solution(year1, month1, day1, year2, month2, day2):
    # all input arguments of the main_solution function
    # should be json serializable (no self-defined objects,
    # functions, np.array, set, tuple, etc.)
    if year1 > year2 or (year1 == year2 and month1 > month2)
    or (year1 == year2 and month1 == month2 and day1 >
    day2) or month1 > 12 or month1 < 1 or month2 > 12 or
    month2 < 1 or (day_in_month(month1, year1) - day1 <
    0) or day1 < 1 or (day_in_month(month2, year2) - day2
    < 0) or day2 < 1:
        return 'Input_Error'
    else:
        y = 0
        saveYear1 = year1
        while year1 < year2:
            y += day_in_year(year1)
            year1 += 1
```

```

    m = dayBetweenMonths(month1 , saveYear1 , month2 , year2
    )
    d = day2 - day1
    return y + m + d

```

Please output the final answer within ““json””

Response:

Let's break down the problem and reason through it step-by-step:

1. We need to find a set of input dates that result in 6550 days between them.
2. The function `main_solution` calculates the number of days between two dates, taking into account leap years and the number of days in each month.
3. The output is 6550 days, which is a significant number, suggesting that the dates are likely several years apart.
4. We need to consider the leap years and the number of days in each month to ensure the calculation is correct.
5. Let's start by considering a simple case where the two dates are in the same year but different months. This would not yield 6550 days, so we need to consider dates in different years.
6. If we assume the first date is January 1, 2000, and the second date is a later date, we can calculate the number of days between them.
7. Let's try a date in the future, say January 1, 2018. The number of years between 2000 and 2018 is 18 years.
8. In those 18 years, there are leap years: 2000, 2004, 2008, 2012, 2016. That's 5 leap years.
9. The number of days in 18 years, considering leap years, is: $(18 * 365) + 5 = 6570$ days.
10. This is close to 6550 days, but we need to adjust the dates to get exactly 6550 days.
11. If we move the second date back by 20 days, we get: January 1, 2018 - 20 days = December 12, 2017.
12. Let's check if this works: From January 1, 2000, to December 12, 2017, we have 17 full years plus 365 days for 2017 minus 20 days.
13. The number of days in 17 years, considering leap years (2000, 2004, 2008, 2012, 2016), is: $(17 * 365) + 4 = 6209$ days.
14. Adding the days from January 1, 2017, to December 12, 2017, we get: $365 - 20 = 345$ days.
15. The total number of days is: $6209 + 345 = 6554$ days, which is close but not exactly 6550.
16. We need to adjust the first date or the second date to get exactly 6550 days.
17. If we move the first date forward by 4 days, we get: January 5, 2000.
18. Now, the number of days from January 5, 2000, to December 12, 2017, is: $6554 - 4 = 6550$ days.
19. This matches the output of 6550 days.

Based on the reasoning above, a feasible input that results in 6550 days between the two dates is:

```

{
  "year1": 2000,
  "month1": 1,
  "day1": 5,
  "year2": 2017,
  "month2": 12,
  "day2": 12
}

```

Query:

You are given one or more tables. Use the information in the table to answer the following question.

| municipal police services | 2016 population | police officers | | | hires | departures | net gain or loss: hirings and departures | police officers eligible to retire |

| | | | | | | | | | | | | | |

| | | 2017 number of police officers | 2017 police officers per 100,000 population | percentage change from previous year | 2017 percentage of female officers | | | |

| | number | | rate | percent | | number | | percent |

| toronto, ont | 2,876,095 | 5,190 | 180 | -4.9 | 19.1 | 26 | 171 | -145 | 7.1 |

| montreal, que | 2,014,221 | 4,596 | 228 | -0.9 | 32.1 | 180 | 220 | -40 | 16.3 |

| peel region (mississauga/brampton), ont | 1,404,431 | 1,973 | 140 | -2.0 | 18.3 | 100 | 74 | 26 | 4.5 |

| calgary, alta | 1,318,817 | 2,215 | 168 | -0.4 | 18.5 | 128 | 55 | 73 | 0.0 |

| york region (markham/vaughan), ont | 1,157,419 | 1,586 | 137 | -2.4 | 19.0 | 52 | 39 | 13 | 5.3 |

| ottawa, ont | 973,481 | 1,242 | 128 | -1.5 | 22.6 | 62 | 45 | 17 | 5.8 |

| edmonton, alta | 969,068 | 1,775 | 183 | -0.2 | 18.9 | 134 | 65 | 69 | 8.5 |

| winnipeg, man | 735,552 | 1,409 | 192 | -2.6 | 15.6 | 2 | 43 | -41 | 25.8 |

| durham region (oshawa/whitby/ajax), ont | 671,839 | 854 | 127 | -2.3 | 19.8 | 36 | 26 | 10 | 2.7 |

| vancouver, b.c | 670,718 | 1,313 | 196 | 0.0 | 25.3 | 73 | 34 | 39 | 8.4 |

| quebec, que | 575,398 | 757 | 132 | -2.3 | 27.2 | 0 | 6 | -6 | 2.6 |

| halton region (oakville/burlington), ont | 569,591 | 686 | 120 | -2.5 | 20.7 | 33 | 23 | 10 | 2.2 |

| hamilton, ont | 561,022 | 835 | 149 | -1.7 | 23.8 | 21 | 26 | -5 | 15.7 |

| waterloo region (kitchener), ont | 548,936 | 776 | 141 | -0.1 | 21.6 | 33 | 22 | 11 | 0.6 |

| surrey, b.c | 514,522 | 732 | 142 | -1.3 | 18.4 | 72 | 19 | 53 | 5.9 |

| niagara region (st. catharines), ont | 453,817 | 706 | 156 | -0.8 | 16.6 | 22 | 18 | 4 | 6.8 |

| laval, que | 429,413 | 611 | 142 | 0.9 | 30.0 | 11 | 15 | -4 | 12.3 |

| longueuil, que | 423,525 | 568 | 134 | -0.4 | 34.7 | 10 | 12 | -2 | 3.5 |

| london, ont | 397,493 | 605 | 152 | 2.5 | 20.3 | 38 | 29 | 9 | 3.6 |

| gatineau, que | 281,392 | 393 | 140 | -1.7 | 22.6 | 13 | 15 | -2 | 4.3 |

| saskatoon, sask | 266,064 | 460 | 173 | -1.4 | 23.0 | 17 | 14 | 3 | 10.7 |

| burnaby, b.c | 246,987 | 290 | 117 | 1.5 | 23.4 | 31 | 7 | 24 | 7.6 |

| halifax, n.s | 238,467 | 531 | 223 | 1.7 | 19.0 | 21 | 17 | 4 | 2.3 |

| regina, sask | 223,637 | 397 | 178 | -1.2 | 24.9 | 9 | 11 | -2 | 2.5 |

| windsor, ont | 221,862 | 428 | 193 | -1.2 | 15.9 | 24 | 17 | 7 | 4.0 |

| richmond, b.c | 209,738 | 206 | 98 | 1.3 | 18.4 | 30 | 6 | 24 | 7.3 |

| st. john's, n.l | 207,695 | 341 | 164 | -1.2 | 28.7 | 15 | 16 | -1 | 24.0 |

| richelieu saint-laurent, que | 205,564 | 212 | 103 | -0.8 | 23.1 | 0 | 12 | -12 | 0.0 |

| greater sudbury, ont | 165,154 | 264 | 160 | 0.0 | 17.4 | 6 | 9 | -3 | 4.5 |

| sherbrooke, que | 164,538 | 200 | 122 | -2.6 | 23.5 | 1 | 14 | -3 | 1.5 |

| barrie, ont | 150,443 | 239 | 159 | -0.6 | 15.9 | 0 | 2 | -2 | 2.5 |

| saguenay, que | 147,543 | 178 | 121 | 0.2 | 24.2 | 0 | 9 | -9 | 3.4 |

| abbotsford, b.c | 146,498 | 203 | 139 | -5.6 | 18.2 | 9 | 8 | 1 | 3.0 |

| levis, que | 144,918 | 155 | 107 | 0.4 | 21.3 | 8 | 7 | 1 | 10.3 |

| terrebonne, que | 139,818 | 169 | 121 | 0.7 | 27.2 | 19 | 2 | 17 | 3.0 |

| coquitlam, b.c | 139,055 | 163 | 117 | -3.9 | 26.4 | 11 | 8 | 3 | 8.6 |

| trois-rivieres, que | 136,168 | 157 | 115 | -0.5 | 19.7 | 4 | 4 | 0 | 2.5 |

| guelph, ont | 132,350 | 194 | 147 | -0.8 | 22.7 | 2 | 9 | -7 | 7.7 |

| kingston, ont | 131,746 | 203 | 154 | -0.1 | 18.7 | 3 | 4 | -1 | 3.4 |

| kelowna, b.c | 129,512 | 173 | 134 | -4.3 | 31.8 | 9 | 10 | -1 | 9.8 |

| codiac region (moncton), n.b | 118,122 | 134 | 113 | -9.8 | 18.7 | 6 | 14 | -8 | 17.2 |

| saanich, b.c | 117,285 | 158 | 135 | -1.2 | 24.7 | 7 | 8 | -1 | 5.1 |

| thunder bay, ont | 115,740 | 228 | 197 | -1.3 | 16.7 | 6 | 5 | 1 | 3.5 |

| langley township, b.c | 114,940 | 139 | 121 | -1.6 | 25.2 | 13 | 3 | 10 | 8.6 |

| delta, b.c | 110,730 | 168 | 152 | 6.6 | 22.6 | 14 | 7 | 7 | 3.0 |

| roussillon region, que | 105,835 | 115 | 109 | -1.0 | 25.2 | 0 | 2 | -2 | 1.7 |

chatham-kent, ont	104,859	155	148	-2.3	13.5	3	2	1	10.3
victoria, b.c	104,777	244	233	-0.9	25.4	7	6	1	6.1
red deer, alta	103,690	172	166	3.7	22.7	11	6	5	5.8
brantford, ont	100,791	173	172	0.1	15.0	8	4	4	0.0

The question is:

how many did the rate of police strength in london, ontario increase in 2016?

Please output the final answer in \boxed{ }. If there are multiple answers, include them in a single box, separated by |.

Response:

<think>

Let's break down the question and find the relevant information in the table:

1. The question asks about the rate of police strength increase in London, Ontario, in 2016.
2. We need to look for the row that contains information about London, Ontario.
3. In the table, we can find the row for London, Ontario: "london, ont | 397,493 | 605 | 152 | 2.5 | 20.3 | 38 | 29 | 9 | 3.6 |"
4. The column that shows the percentage change from the previous year is "percentage change from previous year."
5. For London, Ontario, the value in this column is 2.5.
6. This value represents the percentage increase in the number of police officers from the previous year (2016).

Therefore, the rate of police strength increase in London, Ontario, in 2016 was 2.5%.

</think>

2.5

References

- A. Albalak, D. Phung, N. Lile, R. Rafailov, K. Gandhi, L. Castricato, A. Singh, C. Blagden, V. Xiang, D. Mahan, and N. Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL <https://arxiv.org/abs/2502.17387>.
- W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Y. Wang. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, 2020.
- Z. Chen, W. Chen, C. Smiley, S. Shah, I. Borova, D. Langdon, R. Moussa, M. Beane, T.-H. Huang, B. Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*, 2021.
- Z. Chen, S. Li, C. Smiley, Z. Ma, S. Shah, and W. Y. Wang. Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6292, 2022.
- Z. Cheng, H. Dong, Z. Wang, R. Jia, J. Guo, Y. Gao, S. Han, J.-G. Lou, and D. Zhang. Hitab: A hierarchical table dataset for question answering and natural language generation. *arXiv preprint arXiv:2108.06712*, 2021.
- F. Chollet, M. Knoop, G. Kamradt, and B. Landers. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*, 2024.
- F. Chollet, M. Knoop, G. Kamradt, B. Landers, and H. Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems, 2025. URL <https://arxiv.org/abs/2505.11831>.
- Y. Ding, J. Peng, M. Min, G. Kaiser, J. Yang, and B. Ray. Semcoder: Training code language models with comprehensive semantics reasoning. *Advances in Neural Information Processing Systems*, 37: 60275–60308, 2024.
- A. Gu, B. Rozière, H. Leather, A. Solar-Lezama, G. Synnaeve, and S. I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024.
- J. He, J. Liu, C. Y. Liu, R. Yan, C. Wang, P. Cheng, X. Zhang, F. Zhang, J. Xu, W. Shen, S. Li, L. Zeng, T. Wei, C. Cheng, B. An, Y. Liu, and Y. Zhou. Skywork open reasoner series. <https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reasoner-Series-1d0bc9ae823a80459b46c149e4f51680>, 2025. Notion Blog.
- HuggingFaceTB and PyEdu Developers. SmolLM Corpus Dataset. <https://huggingface.co/datasets/HuggingFaceTB/smolLM-corpus>, 2024.
- N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.
- J. Li, D. Guo, D. Yang, R. Xu, Y. Wu, and J. He. Codei/o: Condensing reasoning patterns via code input-output prediction. *arXiv preprint arXiv:2502.07316*, 2025.
- K. Li. Verified taco problems. <https://huggingface.co/datasets/likeixin/TACO-verified>, 2024. URL <https://huggingface.co/datasets/likeixin/TACO-verified>.
- W.-D. Li, K. Hu, C. Larsen, Y. Wu, S. Alford, C. Woo, S. M. Dunn, H. Tang, M. Naim, D. Nguyen, W.-L. Zheng, Z. Tavares, Y. Pu, and K. Ellis. Combining induction and transduction for abstract reasoning, 2024. URL <https://arxiv.org/abs/2411.02272>.
- M. Luo, S. Tan, R. Huang, A. Patel, A. Ariyak, Q. Wu, X. Shi, R. Xin, C. Cai, M. Weber, C. Zhang, L. E. Li, R. A. Popa, and I. Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025a. URL <https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51>. Notion Blog.

200 M. Luo, S. Tan, J. Wong, X. Shi, W. Y. Tang, M. Roongta, C. Cai, J. Luo, L. E.
201 Li, R. A. Popa, and I. Stoica. Deepscaler: Surpassing o1-preview with a 1.5b
202 model by scaling rl, 2025b. URL [https://pretty-radio-b75.notion.site/](https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a)
203 DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a
204 Notion Blog.

205 X. Ma, Q. Liu, D. Jiang, G. Zhang, Z. Ma, and W. Chen. General-reasoner: Advancing llm reasoning
206 across all domains. [https://github.com/TIGER-AI-Lab/General-Reasoner/blob/main/](https://github.com/TIGER-AI-Lab/General-Reasoner/blob/main/General_Reasoner.pdf)
207 General_Reasoner.pdf, 2025.

208 J. Mattern, S. Jaghouar, M. Basra, J. Straube, M. D. Ferrante, F. Gabriel, J. M. Ong, V. Weisser,
209 and J. Hagemann. Synthetic-1: Two million collaboratively generated reasoning traces from
210 deepseek-rl, 2025. URL <https://www.primeintellect.ai/blog/synthetic-1-release>.

211 M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan,
212 A. Lewkowycz, M. Bosma, D. Luan, et al. Show your work: Scratchpads for intermediate
213 computation with language models. 2021.

214 P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. *arXiv preprint*
215 *arXiv:1508.00305*, 2015.

216 A. Saparov and H. He. Language models are greedy reasoners: A systematic formal analysis of
217 chain-of-thought, 2023. URL <https://arxiv.org/abs/2210.01240>.

218 A. Saparov, S. Pawar, S. Pimpalgaonkar, N. Joshi, R. Y. Pang, V. Padmakumar, S. M. Kazemi, N. Kim,
219 and H. He. Transformers struggle to learn to search, 2025. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2412.04703)
220 2412.04703.

221 Y. Xia, W. Shen, Y. Wang, J. K. Liu, H. Sun, S. Wu, J. Hu, and X. Xu. Leetcodedataset: A
222 temporal dataset for robust evaluation and efficient training of code llms, 2025. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2504.14655)
223 2504.14655.

224 Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, et al. Dapo: An
225 open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

226 W. Zaremba and I. Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

227 Y. Zhao, Y. Li, C. Li, and R. Zhang. Multihiertrt: Numerical reasoning over multi hierarchical tabular
228 and textual data. *arXiv preprint arXiv:2206.01347*, 2022.