Analysis of Mistake Messages

IQP Report

By Wesley Lo

Date

2 December 2021

Acknowledgements

I would like to express my gratitude to Professor Neil Heffernan for offering me the amazing opportunity to work on ASSISTments over the past year. I really appreciate the time he has taken out of his inhumanly busy schedule to get to know me and offer support both throughout the project and in other areas. I am also super grateful to Professor Stacy Shaw for co-mentoring me on this project. She has been an incredibly supportive and positive person to work with. I feel extremely blessed for the opportunity to work with both of them and for generously them taking time each week to meet and offer support. It's been an amazing pleasure get to know them over the past year.

I would also like to thank Kennedy Damoah. He has offered great guidance throughout this project, and I am super grateful for the meetings we've had together. It's been a pleasure to get to know him this year.

Abstract

ASSISTments is a math-based educational platform used by a few hundred researchers and middle school teachers combined (Heffernan, 2014). The platform allows teachers to compose problem sets relevant to class curriculum, while researchers can readily analyze data from students solving the problems. One function of ASSISTments is to generate adaptive, error-dependent messages for students called "mistake messages" or “feedback messages” that students receive directly after making a mistake. In this paper, we describe our process in trying to analyze on a 50-question problem set of 2-step algebraic multiplication and division problems built in ASSISTments, the goal to identify changes in student performance and behavior depending on whether they've received a message for their mistake or not.

Intro/Background

ASSISTments is a math-based educational platform used by a few hundred researchers and middle school teachers combined (Heffernan, 2014). The platform allows teachers to compose problem sets relevant to class curriculum, while researchers can readily analyze data from students solving the problems. In this section, we will first give background on the problem set we worked on and what mistake messages are.

For this IQP (Interactive Qualifying Project), we analyzed 2-step multiplication and division problem set, built in ASSISTments by Dr. Douglas Selent, a former computer science PhD researcher at WPI (Dr. Douglas Selent, n.d.) This problem set has the ID **PSAHQV**, and we will use this code to refer to the problem set throughout the paper. 14265 students worked on this problem set, each randomly assigned to either the treatment or message condition. Each condition has 50 multiplication and division problems (25 each) of the form shown in Table 1, in which a, b, and c are randomly generated integers, such that x is an integer and is to be solved for. Note the problems in the treatment and control groups were generated once, and problems in the treatment group were different than those in the control group. These problems were provided in a randomized order, and students needed to complete 3 of them correctly in a row to complete the problem set.

Division Problem	Multiplication Problem
$\frac{x}{a} + b = c$	$a \cdot x + b = c$

Table 1

The ASSISTments platform allows teachers and researchers to construct feedback messages, which get directly displayed to students after they input a certain answer. In Selent's study, students in the message condition received a feedback message if they input an **expected common wrong answer** (ECWA), while students in the control condition received no feedback messages. Selent worked with students on these problems and accordingly developed these ECWAs, based on his observations about what kinds of mistakes students tended to make most commonly. Selent came up with a set of formulas for both multiplication and division problems that corresponded to these ECWAs and wrote feedback messages for students to provide guidance for them after a common wrong answer. To do this, note that the equations written in Table 1 can each be reconfigured such that x is on one side of the equation and a combination of a, b, and c can be written on the other (Table 2). The ECWAs are simply variations of variable combinations, depending on where the mistake occurs in the 2-step process.

Division Problem	Multiplication Problem
$x = (c - b) \cdot a$	$x = \frac{(c - b)}{a}$

Table 2

PSAHQV was constructed using ASSISTments' variabilized template feature. Variabilized templating is a tool in ASSISTments, built using the programming language "Ruby on Rails," that allows researchers and teachers to generate randomized equations with a desired format (Variabilized Templates, n.d.). In PSAHQV, two templates, one for division and one for multiplication, were used generate the 50 questions. Figure 1 shows what the template equation is for a division problem, while Figure 2 shows the segment of the problem editor that generates the numbers used in the equation.

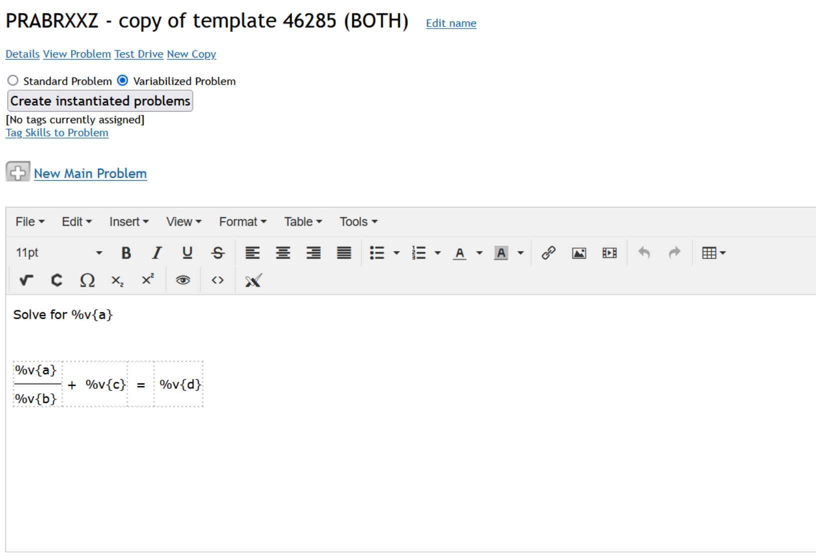


Figure 1 – Problem editor displaying question

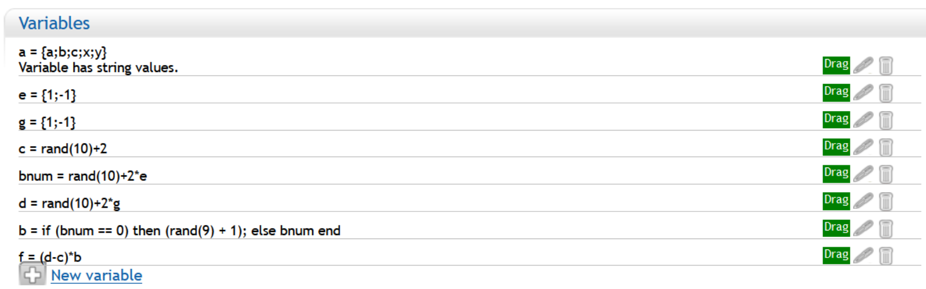
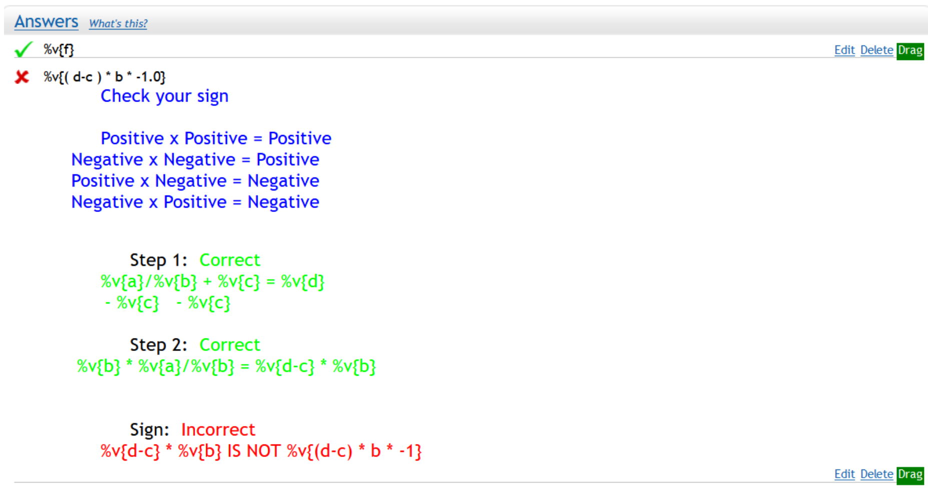


Figure 2 – Problem editor displaying variables

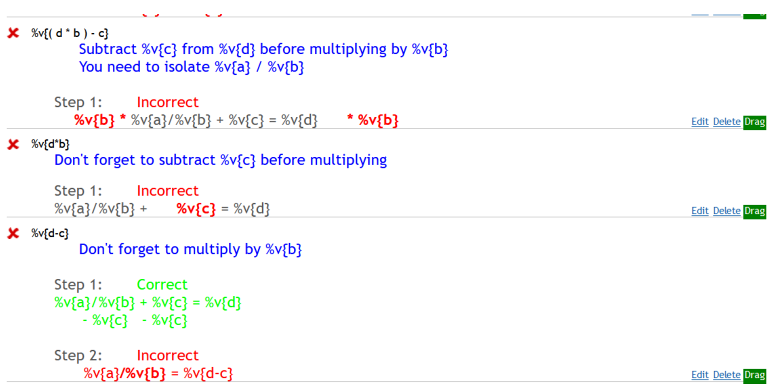
Finally, the feedback message templates, examples of which are provided in Figures 3a and 3b, can display the desired text with customized formatting. As shown in Figure 3a, additional guidance such as showing that a positive number multiplied by a negative number is negative, etc. can be provided in feedback messages, and colors such as red, blue, and green, can be used to indicate correctness or more descriptive feedback.

This problem set was constructed with seven templates for multiplication and eight for division. Examples are provided in the following figures. Examples of all ECWA types are shown in Appendix H.

(a)



(b)



Figures 3a, b – Examples of mistake message templates

Questions in this problem set, and hence in the variabilized templates, also include hints. Students can ask for a hint 3 times. The first hint shows how a similar problem can be solved. Note, red and blue text are here used to help emphasize the steps. The second and third hints show the same steps applied specifically to this problem. The hint template for division is shown below in Figure 4.

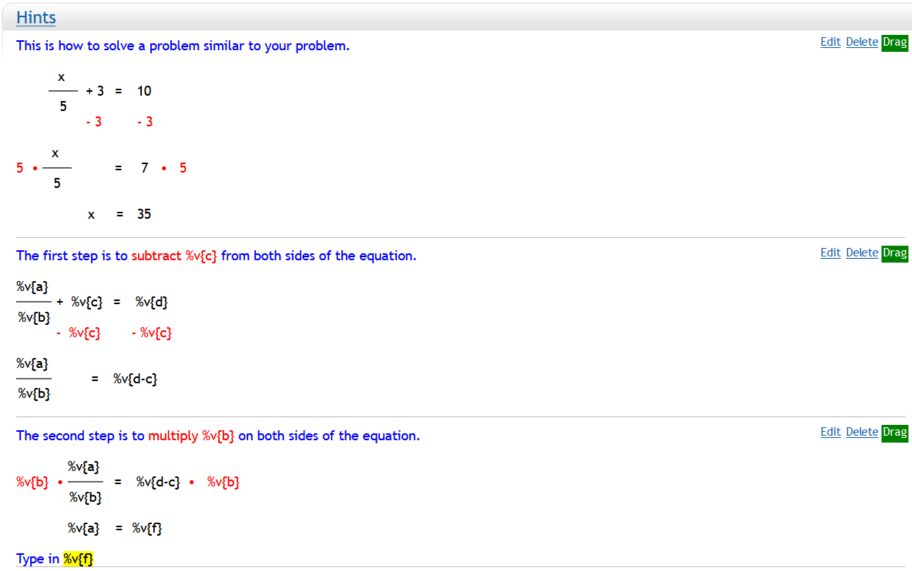
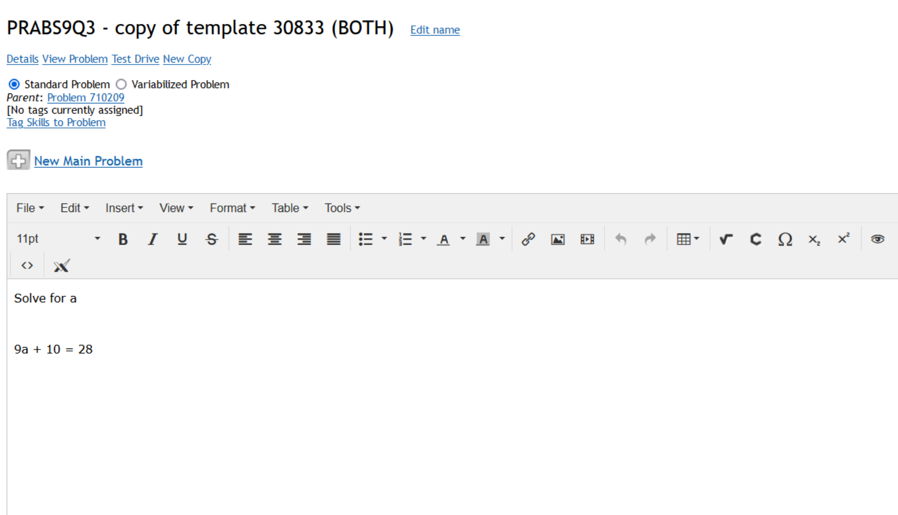


Figure 4 – Example of hint template

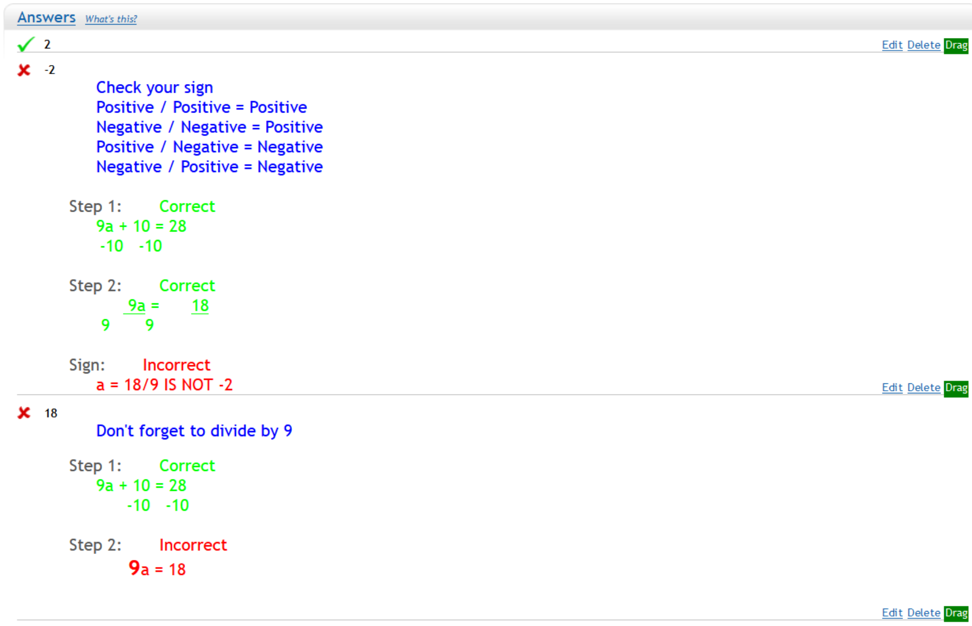
Finally, the problems used in this problem set were then generated using the *Create instantiated problems* button shown in Figure 1, accordingly with randomized feedback messages, hints, and with a solution. Each problem was generated with a problem ID that allowed us to identify them in the various data sources that will be discussed later in this paper.

Figures 5a, b, and c show example screenshots of what a problem generated via the template looks like in the editor menu.

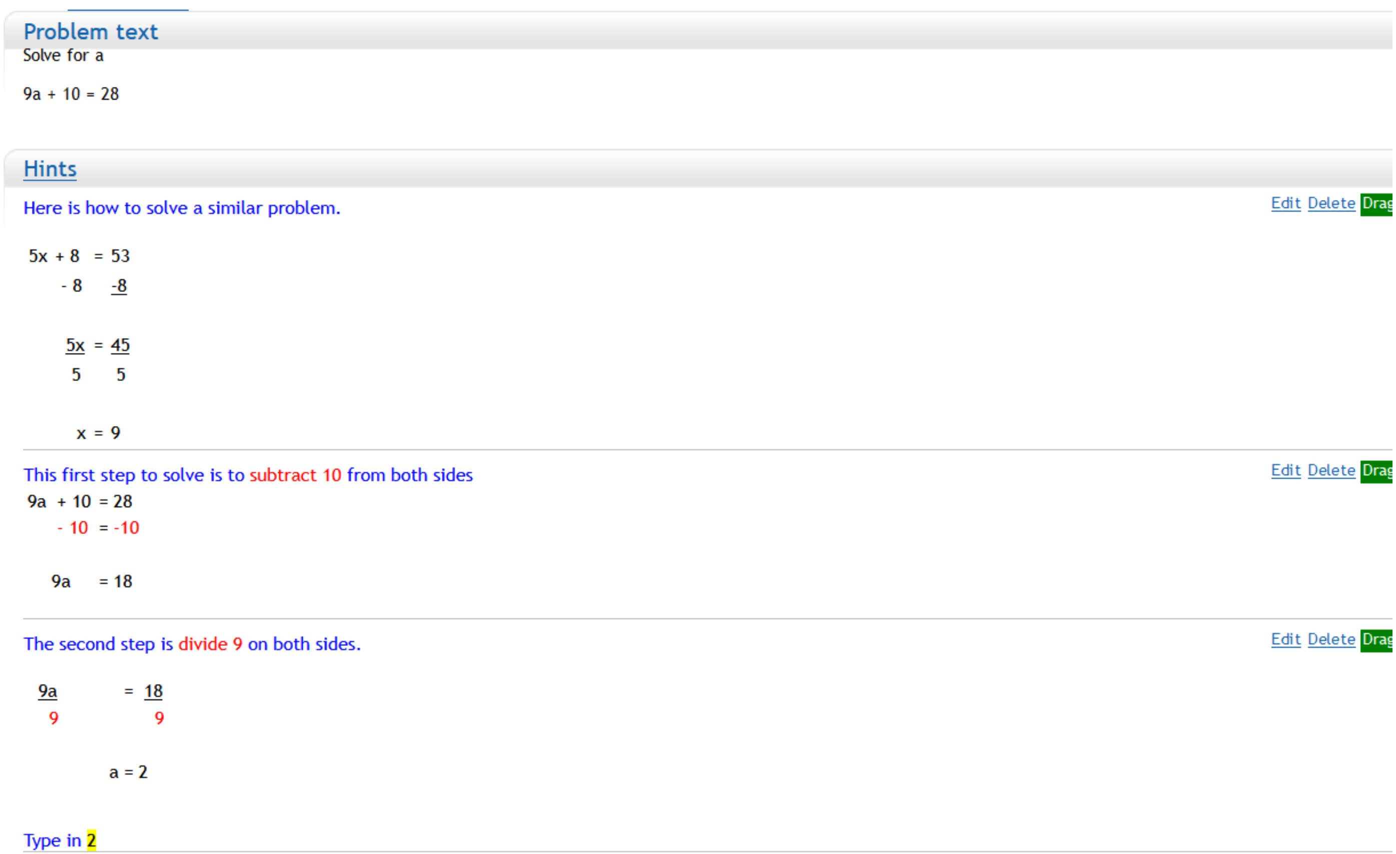
(a)



(b)



(c)



Figures 5a, b, c – Example of editor of generated problem. (a): Question editor (b): Examples of mistake messages (c): Example of hint

Overall, our goal with analyzing data on this problem set was to see what kinds of effects feedback messages have on student behavior and how they perform. Do they help students to perform better on subsequent problems? Do students tend to ask for less hints after receiving a mistake message? Or do students not pay attention to mistake messages at all? Is there a way to measure the effectiveness of these messages? These are all research questions we were hoping to think about through our analysis.

Data scraping + Preprocessing

ALI-Doc Request: Getting + Formatting Student data

Our first step was to submit an ALI-Doc request to get data on the different actions students made when working on the problem set. ALI, Assessment of Learning Infrastructure, is a tool for researchers of the ASSISTments platform that provides data relevant to problems and students completion of them, in the ASSISTments system. The ALI-Doc request provides data sheets in the form of csv (comma separated values) files at the action level, problem level, and student level, as described below in Figure 6. (ALI's Analytics, n.d.)

1. Action Level - One row per action per student; the finest granularity. Students participating in your study have performed 273119 actions (e.g., beginning problems, attempting to answer problems, asking for hints or tutoring, and eventually completing problems).

2. Problem Level - One row per problem per student. Students participating in your study have completed 70323 problems. The flow through a single problem incorporates many actions, resulting in a coarser data file (fewer rows).

3. Student Level - One row per student; the coarsest granularity. Columns are laid out in opportunity order to depict the student�s progression through the problem set. Problem level information is expanded to one column per problem per field (column heavy).

4. Student Level + Problem Level - One row per field per student. Columns are laid out in opportunity order to depict the student�s progression through the problem set. This is an alternative view of the student level information (row heavy).

If after consulting our glossary page you have trouble interpreting any of the above files, please feel free to email assistments-data@wpi.edu

The ASSISTments Research Team

Figure 6 – Information on datasets in ALI-Doc request

Considering we were interested in data relevant to how students answered problems, we started looked at the action level dataset. This level included information such as the problem ID of the problem the student was working on, whether the student requested a hint or submitted an answer (action type), correctness of their answer, what they input as an answer, and the timestamps of their responses. Each row represents such an action a student made on a particular problem.

For our analysis, we settled on Python for its diverse set of prepackaged data analysis tools for parsing, processing, and analyzing data, in addition to flexibility with file manipulation, statistical libraries, and for writing our own functions. All the programming performed for this IQP was written using python and its various libraries (Python, n.d.).

Processing Student Data

One of the first observations we made about the dataset is that there were a lot of concurrent dimensions at play. We were interested in finding a way to format the data in a way convenient for our analysis. Additionally, we wanted a way to synthesize information present at the student, problem, and action levels. We settled on using Python’s class object data structure to store this information.

The idea would be that instead of having to sort through each data table each time, which could potentially be quite slow, we would have info about each student stored in an object that contained relevant information such as the problems they attempted, what actions they performed on what problems, and when those actions occurred. Python objects also allow you to write functions for them, simplifying the amount of data needed to be stored for each student. For instance, instead of storing what their response time was, we could calculate it on an on-need basis.

Data for hints, timestamps, and action types used a dictionary datatype. This means this information could be retrieved using the problem ID. For example, by inputting the problem ID, we would be able to retrieve action timestamps, which could easily give us information about how long it took the student to perform on a particular problem. We stored these dictionaries in a Python class object.

However, this led to having nested data structures, which made it hard to store on the hard drive (and save time instead of having to create these objects every time we wanted to run our analysis), a process called serialization (we used the Python Library, Pickle, for data serialization).

Instead, we decided to convert each object into JSON (JavaScript Object Notation) object, another data format, and then convert them back to python objects when we wanted to run our analysis (JSON, n.d.). We decided to not just use JSON standalone to make the conversion process to a Python object, which allowed for python-specific functions, much more convenient.

In summary, while the processing the data was somewhat time consuming and complicated, this method allowed us to put our data in a more readily analyzable format. Our code for processing the ALI-DOC is in Appendix B, and loading the serialized student data file is provided in Appendix C.

Web scraping: Getting problem values

We noticed that the ALI-Doc request did not include the numbers in each of our 2-step equations, even in the problem level data. Unfortunately, too, the ASSISTments platform lacks an export feature for the equations. Despite this, we wanted to find a way to extract these numbers and in an automated fashion.

The ASSISTments website does provide an option to view all problems of a problem set on a single webpage (See Figure 7). Hence our plan was to select this view option, then download the webpage directly, with the plan to use Python to scrape the site for the equation numbers. Websites are coded using HTML (HyperText Markup Language), so we planned to use the Python library, BeautfulSoup, which is designed to extract data from HTML (HTML: HyperText Markup Language, n.d.) (Beautiful Soup, n.d.).

We first used ASSISTments “view problems” option to print out the whole problem set. A portion of the webpage is shown below in Figure 7.

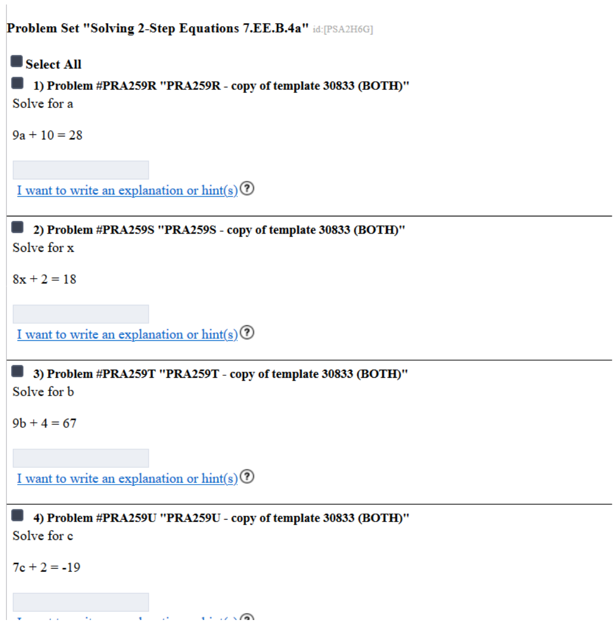


Figure 7 – Examples of a few problems using "view problems" feature

We then downloaded the webpage directly using Chrome's Ctrl-S command.

We then created a script using Beautiful Soup to analyze the HTML. We found that it extracts information in a parent-child manner. To explain this, note that HTML code is generally constructed in a tree-like format. Typically, each line of code has a "tag" that contains information such as text, while it can have "children" that also contain information. Figure 8 depicts an example of an HTML tree. Here the HTML tag contains a body tag, which has tags "h1," "section," and "footer," etc. Here we say "HTML" is a **parent** of "body" and "body" is a **child** of "HTML."

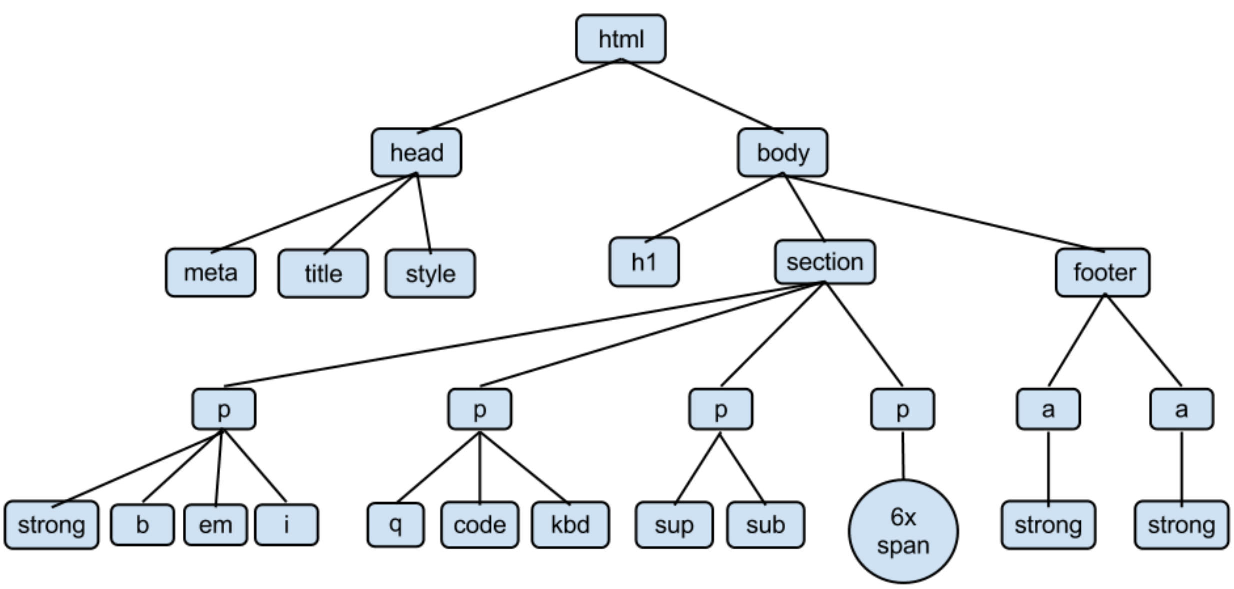
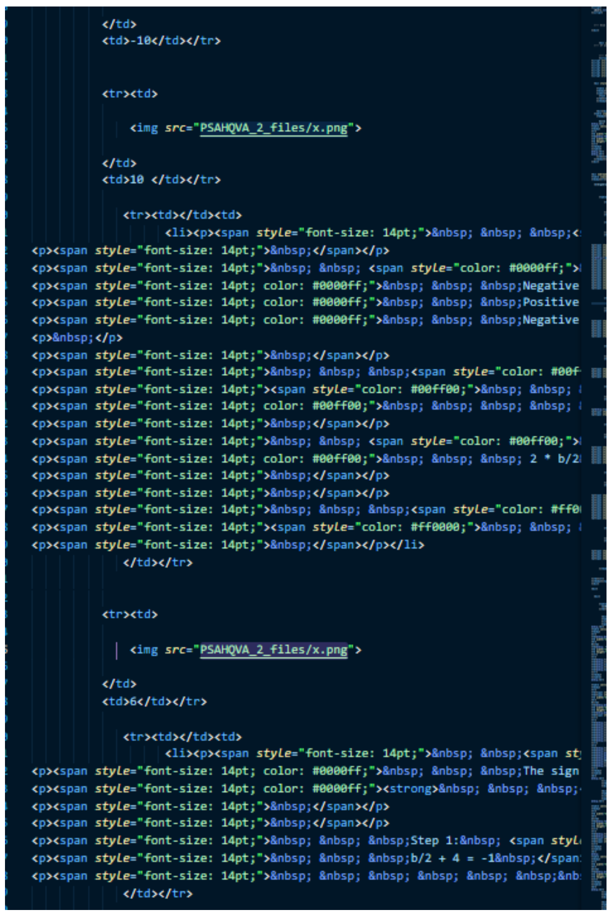


Figure 8 – Example tree diagram of an HTML tree (Lesson 4: The Element Tree, n.d.)

Beautiful Soup can only find specific tags via traversing through the tree. Hence, the process of finding the tags containing information about each problem such as problem title or the equation, involved manually looking through the HTML to identify where these items were.

At first, we ran into a lot of errors because the raw HTML we downloaded from ASSISTments was messy and unformatted. This made it difficult for Beautiful Soup to differentiate which elements were children or parents of which. To combat this, we used html-cleaner which nicely formatted the HTML (HTML Cleaner, n.d.). Seen in Figures 9a and b are the two versions of the same code, pre and post formatting, respectively.

(a)



```
        </td>
        <td>-10</td></tr>


        <tr><td>

          <img src="PSAHQVA_2_files/x.png">

        </td>
        <td>10 </td></tr>

          <tr><td></td><td>
                <li><p><span style="font-size: 14pt;">     <
  <p><span style="font-size: 14pt;"> </span></p>
  <p><span style="font-size: 14pt;">    <span style="color: #0000ff;">
  <p><span style="font-size: 14pt; color: #0000ff;">     Negative
  <p><span style="font-size: 14pt; color: #0000ff;">     Positive
  <p><span style="font-size: 14pt; color: #0000ff;">     Negative
  <p> </p>
  <p><span style="font-size: 14pt;"> </span></p>
  <p><span style="font-size: 14pt;">     <span style="color: #00f
  <p><span style="font-size: 14pt;"><span style="color: #00ff00;">   
  <p><span style="font-size: 14pt; color: #00ff00;">       
  <p><span style="font-size: 14pt;"> </span></p>
  <p><span style="font-size: 14pt;">    <span style="color: #00ff00;">
  <p><span style="font-size: 14pt; color: #00ff00;">      2 * b/2
  <p><span style="font-size: 14pt;"> </span></p>
  <p><span style="font-size: 14pt;"> </span></p>
  <p><span style="font-size: 14pt;">     <span style="color: #ff0
  <p><span style="font-size: 14pt;"><span style="color: #ff0000;">   
  <p><span style="font-size: 14pt;"> </span></p></li>
          </td></tr>


        <tr><td>

          <img src="PSAHQVA_2_files/x.png">

        </td>
        <td>6</td></tr>

          <tr><td></td><td>
                <li><p><span style="font-size: 14pt;">   <span st
  <p><span style="font-size: 14pt; color: #0000ff;">     The sign
  <p><span style="font-size: 14pt; color: #0000ff;"><strong>     
  <p><span style="font-size: 14pt;"> </span></p>
  <p><span style="font-size: 14pt;"> </span></p>
  <p><span style="font-size: 14pt;">     Step 1:  <span styl
  <p><span style="font-size: 14pt;">     b/2 + 4 = -1 </span
  <p><span style="font-size: 14pt;">           &nb
          </td></tr>
```

(b)

```html
<div nobreak="true">
    <table>
        <tbody>
            <tr>
                <td class="clean" width="25px"></td>
                <td>
                    <li>
                        <span style="color: #0000ff;">The second step is to<span style="color: #ff0000;"> multiply 6</span> c
                         <br />

                        <table cellspacing="0" cellpadding="0" border="0">
                            <tbody>
                                <tr>
                                    <td><span style="color: #ff0000;">6 </span></td>
                                    <td><span style="color: #ff0000;"> •</span> </td>
                                    <td width="30" valign="middle" height="10" align="center">
                                        c 
                                        <hr />
                                        6
                                    </td>
                                    <td width="30" valign="middle" height="10" align="center">=</td>
                                    <td height="10"> -6</td>
                                    <td>  <span style="color: #ff0000;">•  </span></td>
                                    <td><span style="color: #ff0000;"> 6</span> </td>
                                </tr>
                                <tr>
                                    <td> </td>
                                    <td> </td>
                                    <td> </td>
                                    <td> </td>
                                    <td> </td>
                                    <td> </td>
                                    <td> </td>
                                </tr>
                                <tr>
                                    <td> </td>
                                    <td> </td>
                                    <td>c</td>
                                    <td>   =</td>
                                    <td>-36</td>
                                    <td> </td>
                                    <td> </td>
                                </tr>
                            </tbody>
                        </table>
                        <br />
                         <br />
                        <span style="color: #0000ff;">Type in </span><span style="background-color: #ffff00; color: #000000;"
                    </li>
                </td>
            </tr>
        </tbody>
```

Figures 9a,b – HTML before (a) and after (b) formatting

With Beautiful Soup now able to analyze the html, we spent a long time and eventually found the specific combination of tags that printed out the problem ID and equation. The process of identifying multiplication versus division problems was the same but parsing the location of the specific numbers was different.

In summary, using our newly created script, we could extract the names of the problem IDs in both conditions and what the numbers were in each of the problems. We saved problems in 4-tuple codes via the following construction: 0 if multiplication or 1 if division, followed by the 3 numbers in the problem. This was saved in a dictionary, with the keys being the problem IDs and the values being these problem 4-tuple "codes".

Finally, we used Pickle to export our dictionary of problems, allowing us to only need to perform this algorithm once. Our code is provided in Appendix A.

Filtering Out Students

With this student and problem data now in a more readily analyzable form, we next wanted to filter out students that we determined to be invalid for our analysis. First, we kept only students who made mistakes (and hence must have received a mistake message) and students who answered questions after 2016, as the problem set hadn't used the mistake messages from 2016 and earlier. To do this, we simply iterated through the students, and checked their "answer timestamps" to see if they occurred after 2016. We used Python's "Datetime" library to do this (Datetime, n.d.).

We also chose to remove students who asked for a hint before making a mistake, as we didn't want the act of receiving a hint before a mistake message to similarly, have the possibility to skew our results. However, this did not lead to any additional students being filtered out. Figure 10 shows the results of this process.

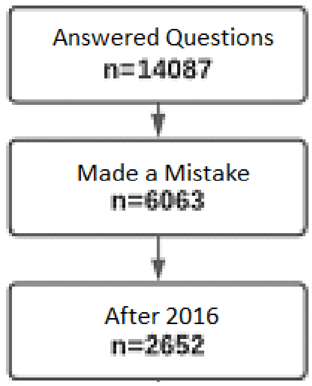


Figure 10 – Number of students at different stages in filtering

Generating Expected Common Wrong Answers

With this filtered set of 2652 students, we next planned on seeing whether their responses were classified as an ECWA, to make sure they received a mistake message in the first place. We also realized that we would be performing a fairer analysis by only comparing students who made a mistake on their first problem.

We chose to only look at first problem for a couple reasons. Firstly, as mentioned earlier, students who get three problems in a row correctly, complete the problem set. So, a significant proportion of our mistake data would come from students making a mistake on their first problem anyways. Secondly, we didn't want a student who had already made a lot of mistakes to be treated by the analysis in the same way that a student who may have gotten a problem right before making a mistake.

We first represented Selent's ECWA formulas for both the multiplication and division problems using python functions as shown in Figure 11.

```python
mistakes_multiplication = [
    lambda x,y,z: (z-y) * x * (-1),
    lambda x,y,z: x * (y+z),
    lambda x,y,z: (z*x) - y,
    lambda x,y,z: z*x,
    lambda x,y,z: z-y,
    lambda x,y,z: z-y-x,
    lambda x,y,z: z*x+y
]

mistakes_division = [
    lambda x,y,z: (z-y) / x,
    lambda x,y,z: z-y,
    lambda x,y,z: (z-y) * x,
    lambda x,y,z: (z+y) / x,
    lambda x,y,z: (z-y-x),
    lambda x,y,z: (y+z),
    lambda x,y,z: ((z-y) / x)+1,
    lambda x,y,z: ((z-y) / x)-1,
]
```

Figure 11 – Python functions used to generate ECWAs

We then iterated through each of the questions represented by the 4-tuples, containing question type and the equation values, and applied these python functions to derive the respective, expected common wrong answers. We stored these in dictionaries. Our code is outlined in Figure 12. Note the full code for this section is provided in Appendix D.

- Iterate through questions in control condition
  - If question is multiplication problem
    - Iterate through ECWA formulas for multiplication
      - Calculate ECWA
      - Save ECWA to dictionary
  - If question is division problem
    - Iterate through ECWA formula for division
      - Calculate ECWA
      - Save ECWA to dictionary
- Repeat for message condition

Figure 12 – Pseudo-code showing ECWA storing

With our ECWAs for each question extracted, we then wrote code to check whether students' first mistakes were expected common wrong answers. To do this, we iterated through each student, iterated through each problem they completed, checked if their first problem was incorrect, then checked if their first mistake was an expected common wrong answer. Our code is outlined as follows in Figure 13:

- iterate through student objects
  - iterate through problem ID's student has worked on
    - if first problem student attempted AND a mistake
      - iterate through student's mistakes
        - parse student's mistake
        - if mistake was a CWA
          - save mistake in a list
          - increment n-students-with-ECWA by one
        - end our search

Figure 13 – Pseudo-code showing how we parsed ECWAs and identified their count

Some students' answers had characters such as parentheses or asterisks. We tried using Python's library, Parser, to extract their answers. We had challenges getting Parser's parsing function to work on their answer inputs, so we wrote a custom function with the help of a Stack Overflow thread, which eventually worked (Evaluating a Mathematical Expression In a String, n.d.).

Finally, we also wanted to determine how many students there were in each group, so we logged how many students had made an expected common wrong answer.

Roadblock: Asymmetric Conditions

We found that while a *similar number* of students in both the treatment and condition groups (1316 and 1336 respectively) and a similar number who made a mistake on their first problem (710 and 724 respectively), there was a *significant discrepancy* in students who made an ECWA in our control versus treatment group (see Figure 14).

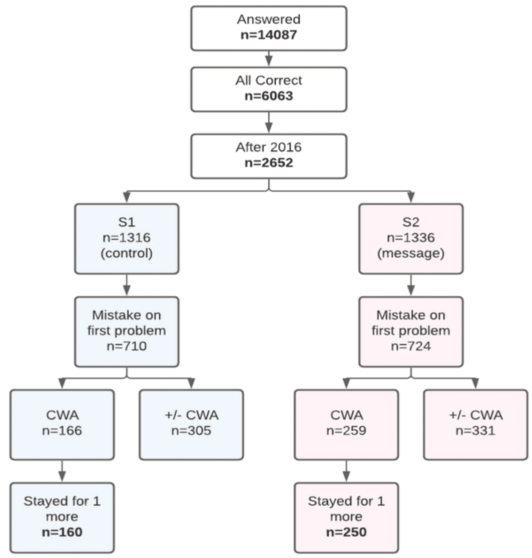


Figure 14 – Number of students at different stages in filtering

Note for this to be a controlled experiment, these two conditions should have been very close in proportion, as there were no intended differences between these two conditions upon the submission of an ECWA. Here, however, 166 people in control condition made an ECWA, and 259 made an ECWA in the message condition. The differences in proportion between the two groups suggested to us an error in the randomization process.

We thoroughly investigated the code to ensure that there weren't any errors in our data processing pipeline causing this discrepancy and found our code to be consistent in its results. We also looked at the ECWAs a bit further. Note that the negative variants of ECWAs were not included in the problem set. For example, if Problem #25 had “-5” as an ECWA, then “5” wouldn’t necessarily be an ECWA, unless it was generated by a separate formula.

As such, we decided to also test the negative variants of ECWAS to see how different the two groups differed (See “+/- CWA” in Figure 14). We found the two groups had much closer proportions of ECWAs on the first problems. This suggested to us that there may have been certain problems that happened to show up in the treatment problem set versus the condition problem set which happened to have negative ECWAs be more common with their specific questions.

At this point, we were uncertain as to the validity of any planned statistical tests on the dataset, given the nature of the differences between these groups.

Improving our Understanding of the Data

New Goals

Despite these challenges in the data, we decided to set new goals and expectations for the project. Firstly, we were interested in making improvements to the design of the study that would be more conducive for an analysis accurate to what we were hoping to measure. Secondly, we hoped to still gain whatever insights we could from the results of the experiment to best proceed with said first goal. Overall, we wanted to see what insights we could learn from our preliminary analysis, while opting to construct an improved experiment.

Improving our Understanding of the Data

First, we wanted to figure out how accurate the mistake messages were, and generally, to extract more problem specific data. We expanded the scope of our previous code (which we used to iterate through the students and determine whether their first mistake was an ECWA) towards iteration towards their other mistakes to find *all* mistakes each student made. We also made sure not to include multiple copies of a given mistake from the same student if they happened to enter it in multiple times. This gave us additional statistics on how difficult problems were and what the top empirical wrong answers, contrasted with the frequency of students making an expected common wrong answer.

Shown below is an example of a figure generated via Python's Matplotlib—a data plotting library—that we configured to display information about problem properties and the frequency of problem types. We generated one graphic for each problem in the dataset, containing information such as problem ID, group (S1: control, S2: treatment), problem accuracy, and the ratio of ECWAs to all kinds of mistakes. We also included what empirical mistakes (all kinds of mistakes) and ECWAs were most frequent to the problem as two separate histograms. All figures are provided in Appendix G.

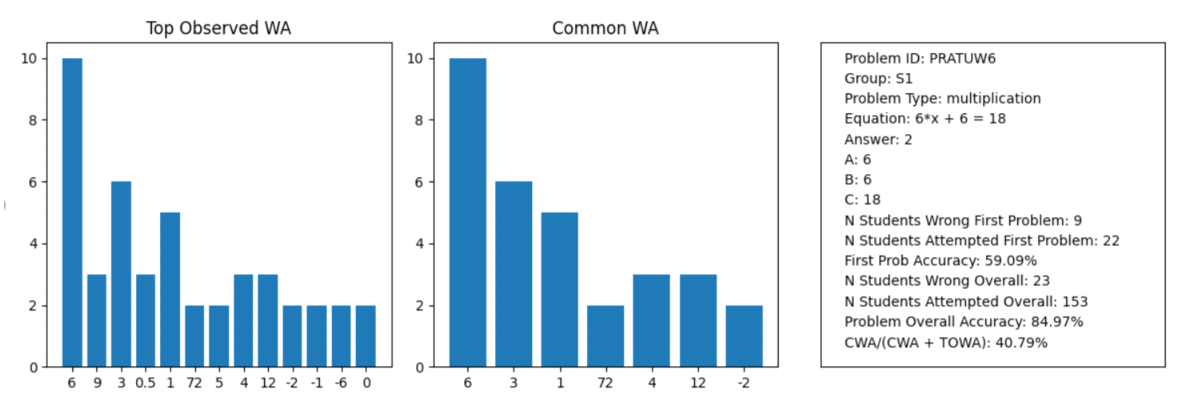


Figure 15 – An example of the problem figures, displaying problem-specific information.

We also generated a spreadsheet, allowing us to easily compare different statistics and properties between different problems. Shown below in Figure 16 is an example of several rows in the sheet. The full spreadsheet is provided in Appendix F.

1	Problem ID	Group		multiplication	Equation	Answer	A	B	C	N Students Wrong First P	N Students Attempted First F	First Prob Accur	N Students Wror	N Students Atter	Problem Ov
2	PRATU5D	S1	2	division	x/7 + 8 = 1	-49	7	8	1	42	49	14.29%	133	211	36.
3	PRATU5R	S1	1	division	x/6 + 11 = 5	-36	6	11	5	28	32	12.50%	93	169	44.
4	PRATU5H	S1	3	division	x/7 + 11 = -1	-84	7	11	-1	34	40	15.00%	93	171	45.
5	PRATU5Z	S1	6	division	x/11 + 9 = 2	-77	11	9	2	22	27	18.52%	91	180	49.
6	PRATU5X	S1	4	division	x/3 + 10 = 2	-24	3	10	2	25	30	16.67%	84	168	50.
7	PRATU52	S1	11	division	x/9 + 7 = 3	-36	9	7	3	24	34	29.41%	80	171	53.
8	PRATU55	S1	23	division	x/3 + 9 = 4	-15	3	9	4	15	27	44.44%	72	162	55.
9	PRATU5Q	S1	17	division	x/7 + 2 = 9	49	7	2	9	17	28	39.29%	68	156	56.
10	PRATU54	S1	9	division	x/4 + 4 = 3	-4	4	4	3	27	35	22.86%	77	181	57.
11	PRATU5E	S1	14	division	x/5 + 6 = 5	-5	5	6	5	20	29	31.03%	66	157	57.
12	PRATU48	S1	5	division	x/6 + 8 = 4	-24	6	8	4	14	17	17.65%	59	141	58.
13	PRATU5K	S1	7	division	x/6 + 5 = 3	-12	6	5	3	26	33	21.21%	75	183	59.
14	PRATU5V	S1	18	division	x/7 + 5 = 0	-35	7	5	0	18	30	40.00%	66	170	61.
15	PRATU5Y	S1	8	division	x/11 + 9 = 7	-22	11	9	7	17	22	22.73%	65	172	62.
16	PRATU5P	S1	13	division	x/10 + 5 = 3	-20	10	5	3	14	20	30.00%	59	158	62.
17	PRATU5F	S1	21	division	x/8 + 2 = 4	16	8	2	4	16	28	42.86%	62	168	63.
18	PRATU5T	S1	20	division	x/2 + 5 = 7	4	2	5	7	17	29	41.38%	64	176	63.
19	PRATU5A	S1	24	division	x/8 + 7 = 6	-8	8	7	6	15	27	44.44%	54	161	66.

Figure 16 – Example rows from our data spreadsheet

Using these two data presentation forms, we were able to deduce insights into the problems. The code used to generate both can also be found in Appendix D.

Insights + Findings

To preface, many of these observations are more qualitative and general. Our intent was less to derive specific results and more to inform the modifications to the problem set we were interested in making as part hypothesis and part minimizing potential noise in new data.

What stood out to us first was the average accuracy on division problems were overall much lower than for multiplication problems (59.75% accuracy vs 75.23% accuracy respectively), indicating a significant difference in difficulty.

In reference to the asymmetry between both conditions, we noticed different values present in equations from our control and message conditions. For example, there were several problems that had -1 as the first value in the no message condition, but no coefficients had -1 in the message condition. However, upon further investigation, we found the variabilized templates used for division and multiplication in both the control and message conditions to be the same. We suspect there may have been significant enough variability in how messages were generated in general, that resulted in differences in which ECWAs were recognized.

We also noticed a few qualitative trends in problems of lower accuracy in the division and multiplication categories. Problems in both division and multiplication with the c term smaller than the b term tended to have a higher difficulty. Our guess for why this was a logical pattern was because the first step being to subtract b from c in both steps, meant that if b > c, then (c-b) would be negative, which may be easier to make a mistake with. Problems with their "a" term equaling 1 generally had a higher accuracy, which we suspect was because this rendered them to really be 1-step addition/subtraction problems. Problems with all numbers positive also tended to be easier, as well as problems that involved division or multiplication by a factor of 5.

While a further, more rigorous/quantitative delve into problem difficulty remains to be a topic to be explored in more depth in future analysis, these heuristics informed the construction of our new problem set as to be later discussed.

Lastly, we were interested in figuring out how representative Selent's messages were. Based on the problem figures, we found that the ECWAs did a good job of generalizing the most common mistakes students tended to make, despite there being many uncommon random unique mistakes that were unaccounted for. We only found two main ways the ECWAs did not cover the most common types of mistakes: Firstly, there were instances where the negative variants of the ECWAs were commonly made by students. However, this tended to be on problems that had multiple of the same numbers. Second, in division problems, students often divided both sides, and with the "a" term in the numerator.

Constructing a New Study

Note: For brevity, note that the original problem set will be referred to as 1.0, while the new problem set will be referred to as 2.0.

Changes to the problems

We next began developing the new problem set. One limitation of the initial messages was a way to figure out what components were useful for students. We were curious as to whether messages containing sentiment, specifically positive and encouraging, would offer any measurable benefit towards students using the platform. Thus, we first wanted to expand our analysis to more clearly delineate between messages with positive sentiment versus neutral. Examples of statements we used in our new messages with positive sentiment are provided below in Figure 17:

"Your thinking about this problem is good,"
"Almost there,"
"These problems can be tough,"
"Whoops,"
"A lot of students make a mistake on this problem,"

Figure 17 – Examples of statements we used in messages with sentiment

Thus, we redesigned the 2.0 problem set with three conditions: (1) No messages, (2) message, and (3) message + positive sentiment. We wanted both conditions with messages to be as close to the same as possible to prevent any subtle differences from skewing our results. Hence, we redesigned the messages using a more neutral approach: We changed the red text to a neutral gray. Some of the messages in 1.0 used capitalization in phrases such as something like "X is NOT Y" which we uncapitalized. We simplified the explanations to just show the step they erred on.

We also added lines "It looks like this first step you probably did correctly" and "but we're guessing this last step you might have made an error" in blue, in between steps for two main reasons: to indicate to students our messages were not necessarily 100% accurate and so both message conditions would transition between steps more in a way a human verbally might.

As for how conditions (2) and (3) differed, we also included segments at the beginning of messages for the sentiment condition, as shown in Figure 18. We used orange text for these messages with the intent to have them be visually different from the rest of the message and being emotionally neutral. Examples of all 2.0 ECWA types are shown in Appendix I.

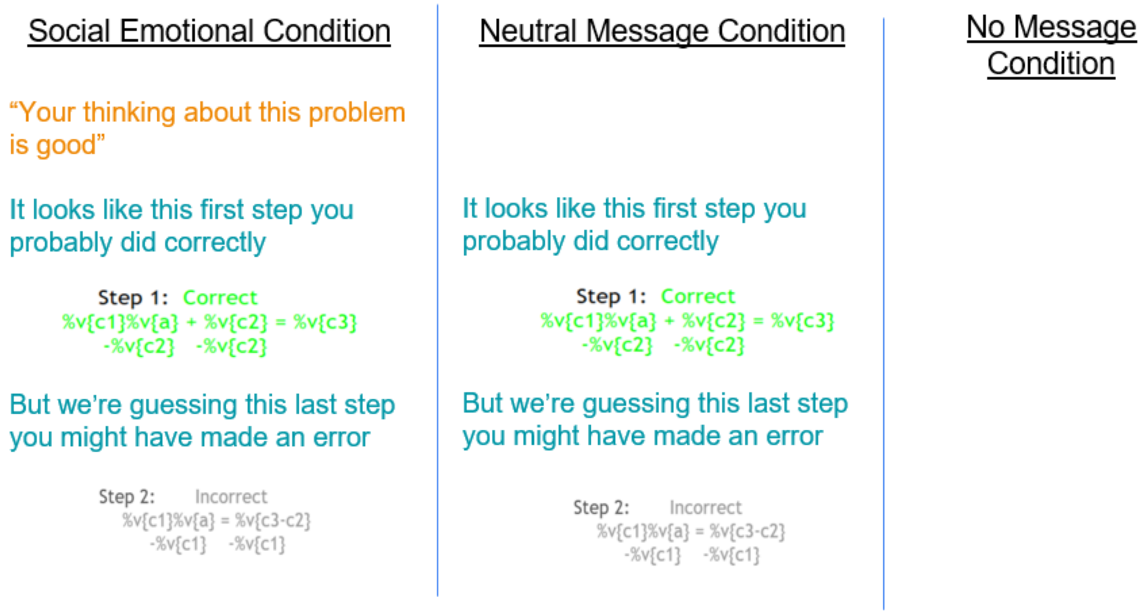


Figure 18 – Structure of messages in each condition

Lastly, we included the new division mistake message described in *Insights + Findings*.

New Problem Set Structure

In 2.0, students first complete one of three unique problems of similar difficulty. Depending on which of the three conditions they are in, they will or will not receive a message with or without sentiment. Afterwards, all students complete the same immediate posttest problem, also of similar difficulty (see Figure 19). Then they continue working on problems from the rest of the problem set, until they get three correct (not including A/B/C or X) in a row.

Our new focus on analyzing only the first mistake students make, motivated this new structure--the idea to get more data and less noise from selecting a narrower set of problems all students first complete. We reused problems from the 1.0 problem set message condition, since they already have the most data, and we've already measured student accuracy on them, and hence can determine problems of similar difficulty. We wanted 3 problems of similar difficulty in case any one problem would be an outlier. We chose the immediate posttest (problem X) to be similar to problems A, B, and C, the idea to increase the chances the same mistake type occurring on the next problem, and hence have more data to compare in occurrences of consecutive mistake types.

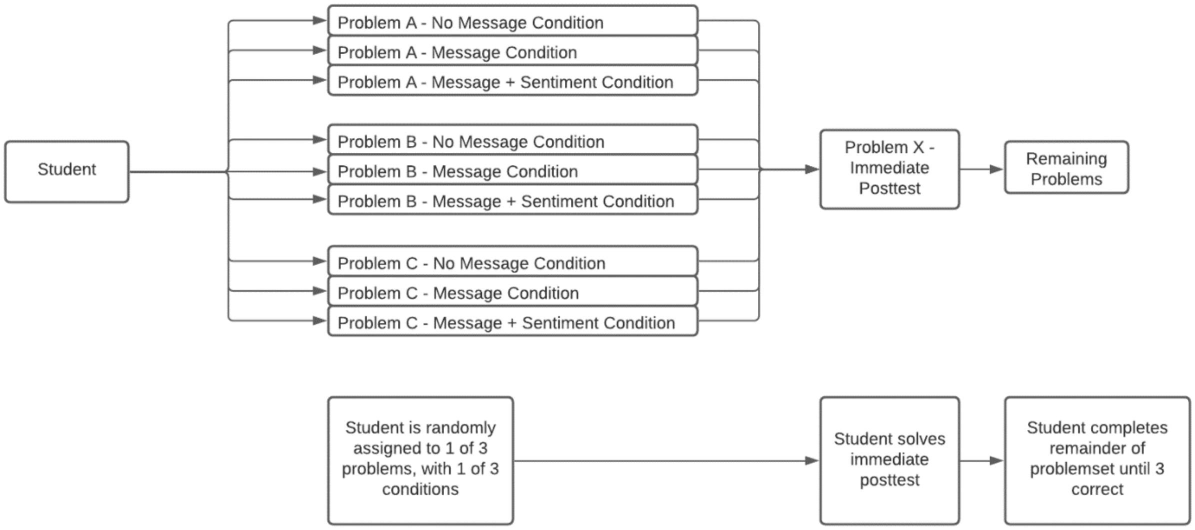


Figure 19 – Diagrams the structure of the new problem set

Choosing First Problems (A, B, C, X)

We used the spreadsheet to select these four problems, shown in Figure 20. In orange are the 3 possible problems students could receive, while in light blue is the next problem they complete. These problems were also chosen from the initial message condition since we already have data for those problems and ECWAs (and note they were not included in the remaining problems). Additionally, we wrote a Python script to ensure the mistake messages did not overlap (code provided in Appendix E). Using our hypothesized heuristics of problem similarity described in *Insights + Findings*, we also considered the a, b, c relation described in the previous section to pick problems we hypothesized to have similar ECWA distributions. Here, all problems share the following properties: they are division problems, b > c, all numbers are positive, coefficients dividing x greater than 1.

Problem ID	Group	Type	Equation	Answer	A	B	C	Problem Overall Accuracy
PRA26BC	S2	division	x/6 + 9 = 3	-36	6	9	3	52.73%
PRA26BU	S2	division	x/10 + 3 = 1	-20	10	3	1	56.36%
PRA26B6	S2	division	x/2 + 6 = 1	-10	2	6	1	57.14%
PRA26BM	S2	division	x/6 + 4 = 3	-6	6	4	3	58.28%

Figure 20 – Subsection of spreadsheet with problems A, B, C, and X

While performance on the first two problems of the problem set is what we will primarily be analyzing, including the remaining problems allows the problem set to function as normal for students, and more data to be collected. We left their messages in 1.0 format.

We have recently launched the 2.0 version of this problem set. It is currently running in ASSISTments, and we hope to collect data over the coming year.

Future Analysis + Ideas for Problem Set 3.0

After we get more data on 2.0, we hope to run statistical tests on how students perform on their next action and posttest, in addition to how their behaviors change dependent on which condition they were in. For instance, how average response time may differ, correctness, how many hints they request, if they received a message with or without sentiment.

We are also interested in incorporating and expanding upon our insights from this analysis. This might include measuring problem difficulty based on the a, b, c relations or analyzing the ECWA type distributions of different problems to find similarities in problem difficulty and whether mistake messages may be more useful for certain problems than others.

We would also like to develop a 3.0 version of the problem set. We've discussed different ways in which problems can be similar or different from one another and generating completely new problems according to different conditions may be interesting to test. This would allow us to further test how effective mistake messages are based on difficulty.

We've also considered using information such as student performance on previous problem sets, as provided in the ALI-Doc data tables, in a predictive model such as logistic regression or random forest to see if types of students—based on trends in their action behavior and general statistics—tend to be effect by mistake messages in a similar way.

Overall, there are many different directions we hope to take this project as we collect more data.

Conclusion

In this paper, we've described our process in analyzing a 50 question 2-step multiplication and division problem set built in ASSISTments. We've built a pipeline that exports student and problem data from the ALI-Doc request and ASSISTments problem builder into an analyzable format. When preparing for our analysis, we suspected problems in the randomization of the initial study, and thus decided to focus on making modifications to the problem set, while learning what we could from the existing data. We made changes to the problem set structure and mistake messages. With these new changes fully implemented, we've officially launched the 2.0 version of the problem set, which is live at the time of this report being written (December 2021). We have described our ideas for future directions for this project, which, as we continue collecting data for the new problem set, we hope to implement.

References

(n.d.). Retrieved from Python: https://www.python.org/

(n.d.). Retrieved from JSON: https://www.json.org/json-en.html

(n.d.). Retrieved from Beautiful Soup: https://beautiful-soup-4.readthedocs.io/en/latest/

(n.d.). Retrieved from HTML Cleaner: https://html-cleaner.com/

ALI's Analytics. (n.d.). Retrieved from ASSISTments Test Bed: https://sites.google.com/site/assistmentstestbed/4-analyze-data/alis-analytics

Datetime. (n.d.). Retrieved from Python: https://docs.python.org/3/library/datetime.html

Dr. Douglas Selent. (n.d.). Retrieved from University of Wisconsin Platteville: https://www.uwplatt.edu/profile/selentd

Evaluating a Mathematical Expression In a String. (n.d.). Retrieved from Stack Overflow: https://stackoverflow.com/questions/2371436/evaluating-a-mathematical-expression-in-a-string

Heffernan, N. T. (2014). The ASSISTments Ecosystem: Building a Platform that Brings Scientists and Teachers Together for. *International Journal of Artificial Intelligence in Education.*, 1.

HTML: HyperText Markup Language. (n.d.). Retrieved from Mozilla: https://developer.mozilla.org/en-US/docs/Web/HTML

Lesson 4: The Element Tree. (n.d.). Retrieved from Open Book Project: http://www.openbookproject.net/tutorials/getdown/css/lesson4.html

Variabilized Templates. (n.d.). Retrieved from Assistments: https://sites.google.com/site/assistmentsadvancedbuilder/variablized-templates

Appendix A: Web Scraping

```
from bs4 import BeautifulSoup
import pandas as pd
import pandas as pd
import dill as pk
import gzip
from sympy import sympify

FILEPATH_PROBLEM_LEVEL = '../data/PSAHQV-02-01-2021-13-26-42-ProblemLevel.csv'
FILEPATH_HTML = '../data/s2_questions.html'
FILEPATH_OUTPUT = '../export/'
F_NAME_QUESTIONS = 'questions_S2.p.gzip'

df_prob = pd.read_csv(FILEPATH_PROBLEM_LEVEL)
with open(FILEPATH_HTML) as f:
    soup = BeautifulSoup(f, "html.parser")

#extracts problem name from html line
def parse_problem_name(html_line):
    return html_line.text.split("#")[1].split(' ')[0]

#extracts text from html line
def parse_html_line(html_line):
    return html_line.text

def parse_question(question):
    q = question.find_all('td')[1].prettify(formatter=lambda s:
s.replace(u'\xa0', ' ')).split('\n')[4]
    if not 'table' in q:
        q = q.split(' ')
        k1 = int(q[1][:-1])
        k2 = int(q[3])
        k3 = int(q[5])
        return (0,k1,k2,k3)
    else:
        q = question.find_all('td')[1].prettify(formatter=lambda s:
s.replace(u'\xa0', ' ')).split('\n')
        k1 = int(q[10].replace(' ',''))
        k2 = int(q[13].replace(' ','').replace('+',''))
        k3 = int(q[19].replace(' ',''))
        return (1,k1,k2,k3)

def clean(s):
    remove_substrings = [u'\xa0', u'\xa0', u'\xa0',u'\n',u'/n']
    for sub in remove_substrings:
```

```python
        s = s.replace(sub,'')
    return s

#gets mistakes and mistake messages from div chunk
def parse_mistake_messages(mistake_messages):

    #find where the mistakes headers are in the html file
    tr_tags = mistake_messages.find('tr').find_all('tr')
    tr_tags = [tr_tags[1]] + tr_tags[3:len(tr_tags):2]
    mistakes = [str(sympify(tr_tag.find_all('td')[1].text, evaluate=True))
                 for i,tr_tag in enumerate(tr_tags)]

    #find where the mistake messages are for mistake
    mistake_messages = mistake_messages.find_all('li')
    mistake_message_dict = {}
    for i,mistake_message in enumerate(mistake_messages):
        steps = [list(map(lambda s: clean(s), [str(message)]))[0]
                for message in mistake_message.find_all('p')]
        #save list of mistake messages
        mistake_message_dict[mistakes[i]] = steps

    return mistake_message_dict

mistake_messages = {}
problems = soup.find_all('div',{'style':'border-bottom: solid; border-width:
1px;'})
question_dict = {}
for j,problem in enumerate(problems):
    problem_name = parse_problem_name(problem.find('font',{'class':'header'}))
    html_problem_sections = problem.find_all('div',{'nobreak':'true'})
    question = parse_question(html_problem_sections[0])
    question_dict[problem_name] = question
print(question_dict)

with gzip.open(FILEPATH_OUTPUT+F_NAME_QUESTIONS,'wb') as f:
    pk.dump(question_dict,f)
```

Appendix B: ALI-Doc Parsing

```
import tqdm
import pandas as pd
import tqdm
import pandas as pd
import dill as pk
import gzip
import datetime
from datetime import datetime
import json

FILEPATH_PROBLEM_LEVEL = '../data/PSAHQV-02-01-2021-13-26-42-ProblemLevel.csv'
FILEPATH_ACTION_LEVEL = '../data/PSAHQV-02-01-2021-13-26-42-ActionLevel.csv'
FILEPATH_STUDENT_LEVEL =
'../data/PSAHQV-02-01-2021-13-26-42-StudentLevelWithScaffolds.csv'
FILEPATH_HTML = '../data/PSAHQVA_formatted.html'
FILEPATH_OUTPUT = '../export/'
F_NAME_MISTAKE_MESSAGES = 'mistake_messages.p.gzip'
F_NAME_STUDENTS = 'students.json'

with gzip.open(FILEPATH_OUTPUT+F_NAME_MISTAKE_MESSAGES,'rb') as f:
    mistake_message_dict = pk.load(f)

df_prob = pd.read_csv(FILEPATH_PROBLEM_LEVEL,low_memory=False)
user_ids = sorted(list(set(df_prob['User ID'])))

df_action = pd.read_csv(FILEPATH_ACTION_LEVEL,low_memory=False)
problem_ids = pd.Series(list(map(lambda x: '"%s"'%(x),
list(mistake_message_dict.keys()))))

def rem_quotes(s):
    return s.replace('"','')

def parse_time(s):
    return datetime.strptime(s, '%m/%d/%Y %H:%M:%S.%f')

class Student:
    def __init__(self,user_id='',
        problem_ids=[],
        correct_answers={},
        mistakes={},
        n_mistakes={},
        messages={},
        n_messages={},
        starts={},
        ends={},
        total_times={},
```

```
        action_orders={},
        action_timestamps={},
        n_hints={},
        hint_timestamps={},
        answer_timestamps={}):

        self.user_id=user_id
        self.problem_ids=problem_ids
        self.correct_answers=correct_answers
        self.mistakes=mistakes
        self.n_mistakes=n_mistakes
        self.messages=messages
        self.n_messages = n_messages
        self.starts=starts
        self.ends=ends
        self.total_times=total_times
        self.action_orders=action_orders
        self.action_timestamps=action_timestamps
        self.n_hints=n_hints
        self.hint_timestamps=hint_timestamps
        self.answer_timestamps=answer_timestamps

def encode_student(s):
    return {'user_id':s.user_id,
            'problem_ids':s.problem_ids,
            'correct_answers':s.correct_answers,
            'mistakes':s.mistakes,
            'n_mistakes':s.n_mistakes,
            'messages':s.messages,
            'n_messages':s.n_messages,
            'starts':DictEncoder(s.starts,DatetimeEncoder),
            'ends':DictEncoder(s.ends,DatetimeEncoder),
            'total_times':s.total_times,
            'action_orders':s.action_orders,
            'action_timestamps':DictEncoder(s.action_timestamps,DatetimeEncoder),
            'n_hints':s.n_hints,
            'hint_timestamps':DictEncoder(s.hint_timestamps,DatetimeEncoder),
            'answer_timestamps':DictEncoder(s.answer_timestamps,DatetimeEncoder)}

def DictEncoder(d,encoder):
    new_d = {}
    for key in d:
        if type(d[key]) == list:
            tmp = []
            for item in d[key]:
                tmp.append(encoder(item))
            new_d[key] = tmp
        else:
```

```python
            new_d[key] = encoder(d[key])
    return new_d

def DatetimeEncoder(d):
    return d.strftime('%m/%d/%Y %H:%M:%S.%f')

def unique_items(l):
    j = []
    for item in list(l):
        if item not in j:
            j.append(item)
    return j

def df_rows(df,col_name,name,neq=False,isin=False):
    if not isin:
        if neq == False:
            return df[df[col_name] == name]
        else:
            return df[df[col_name] != name]
    else:
        return df[df[col_name].isin(name)]

def df_sort(df,col_name,ascending=True,):
    return df.sort_values(col_name,ascending=ascending)

students = {}
event_ids = dict(zip(sorted(list(set(list(map(lambda x:
x.replace('"',''),list(df_action['Action Type'])))) + ['next','work'])),

list('0123456789abcdefghijklmnop')[0:len(set(df_action['Action Type']))+2]))

s_n_problems = {}
s_n_actions = {}

#iterate through users
for i,s_id in tqdm.tqdm(enumerate(user_ids)): #iterate through students
    student_is_old = False

    #subset containing ONLY given user
    s_problem_level = df_rows(df_prob,'User ID', s_id)
    s_action_level = df_rows(df_action,'User ID', s_id)

    #count how many rows in each dataframe
    s_n_problems[s_id] = s_problem_level.shape[0]
    s_n_actions[s_id] = s_action_level.shape[0]


    #store data about student
```

```
    #assumption: student has completed this problem
    s_problem_ids = [] #list of problem ids
    s_correct_answers = {} #correct answer given the problem
    s_mistakes = {} #list of mistakes student made
    s_n_mistakes = {}
    s_messages = {} #dict of messages student received
    s_n_messages = {}
    s_starts = {} #when student started given problem
    s_ends = {} #when student ended given problem
    s_total_time = {} #end - start

    s_action_order = {} #sequence of hints and answers as s or h
    s_action_timestamps = {} #all response times for a given problem

    s_n_hints = {} #how many hints the student asked for on the given problem
    s_hint_timestamps = {} #how many hints the student asked for on the given
problem

    s_answer_timestamps = {}

    #get problem ids in order of time
    df_starts = df_rows(s_action_level,'Action Type','"start"')
    df_starts = df_sort(df_starts,'Timestamp')
    s_problem_ids = list(unique_items(df_starts['Problem ID']))
    rem_ids = []
    for j,p_id in enumerate(s_problem_ids):

        #create subset of df for problem
        df_problem = df_rows(s_action_level,'Problem ID',p_id)
        df_problem = df_sort(df_problem,'Timestamp')

        #student's answers
        df_answers = df_rows(df_problem,'Action Type','"answer"')
        p_answer_timestamps = list(map(lambda x: parse_time(rem_quotes(x)),
list(df_answers['Timestamp'])))

        #answers
        p_mistakes = list(df_rows(df_answers,'Correctness','false')['Answer
Text'])
        p_correct_answer = df_rows(df_answers,'Correctness','true')['Answer
Text']

        #skip problem if there is missing data
        conds = [
            df_rows(df_problem,'Action Type','"start"')['Timestamp'].shape[0] ==
0,
            df_rows(df_problem,'Action Type','"end"')['Timestamp'].shape[0] == 0,
            df_rows(df_problem,'Timestamp','""').shape[0] != 0,
```

```
            df_problem.shape[0] == 0,
            df_answers.shape[0] == 0,
            p_correct_answer.shape[0] == 0
        ]
        if True in conds:
            rem_ids.append(p_id)
            continue

        #basic time data; when problem started, stopped
        p_start = parse_time(rem_quotes(df_rows(df_problem,'Action
Type','"start"')['Timestamp'].iloc[0]))
        p_end = parse_time(rem_quotes(df_rows(df_problem,'Action
Type','"end"')['Timestamp'].iloc[0]))
        p_total_time = (p_end - p_start).total_seconds()

        #timestamps of hints
        p_hint_timestamps = list(map(lambda x: parse_time(rem_quotes(x)),
list(df_rows(df_problem,'Action Type','"hint"')['Timestamp'])))
        p_n_hints = len(p_hint_timestamps)

        #actions and their timestamps
        p_action_order = list(df_rows(df_problem,'Action Type',['"hint"',
'"answer"'],isin=True)['Action Type'])
        p_action_timestamps = list(map(lambda x: parse_time(rem_quotes(x)),
list(df_rows(df_problem,'Action Type',['"hint"',
'"answer"'],isin=True)['Timestamp'])))

        #add mistake messages
        p_messages = {}
        for i,mistake in enumerate(p_mistakes):
            if p_id.strip('"') in mistake_message_dict: #if problem has mistake
messages
                if mistake.strip('"') in mistake_message_dict[p_id.strip('"')]:
#if this kids answer yields a mistake message
                    p_messages[mistake.strip('"')] =
mistake_message_dict[p_id.strip('"')][mistake.strip('"')]

        #save data
        s_correct_answers[p_id] = p_correct_answer.iloc[0] #correct answer given
the problem
        s_mistakes[p_id] = p_mistakes #list of mistakes student made
        s_n_mistakes[p_id] = len(p_mistakes)
        s_messages[p_id] = p_messages #dict of messages student received
        s_n_messages[p_id] = len(p_messages)
        s_starts[p_id] = p_start #when student started given problem
        s_ends[p_id] = p_end #when student ended given problem
        s_total_time[p_id] = p_total_time #end - start
```

```
        s_action_order[p_id] = p_action_order #sequence of hints and answers as s
or h
        s_action_timestamps[p_id] = p_action_timestamps #all response times for a
given problem
        s_n_hints[p_id] = p_n_hints #how many hints the student asked for on the
given problem
        s_hint_timestamps[p_id] = p_hint_timestamps #how many hints the student
asked for on the given problem
        s_answer_timestamps[p_id] = p_answer_timestamps

        #if student not from selent's study
        if s_answer_timestamps[p_id][0].year <= 2016:
            student_is_old = True
            break

    #if student not from selent’s study
    if student_is_old: continue

    for id_ in rem_ids:
        s_problem_ids.remove(id_)
    if len(s_problem_ids)==0: continue

    #save student as Student object and turn object into json
    students[s_id] = json.dumps(Student(
        user_id=s_id,
        problem_ids=s_problem_ids,
        correct_answers=s_correct_answers,
        mistakes=s_mistakes,
        n_mistakes=s_n_mistakes,
        messages=s_messages,
        n_messages=s_n_messages,
        starts=s_starts,
        ends=s_ends,
        total_times=s_total_time,
        action_orders=s_action_order,
        action_timestamps=s_action_timestamps,
        n_hints=s_n_hints,
        hint_timestamps=s_hint_timestamps,
        answer_timestamps=s_answer_timestamps
    ),
    default=encode_student)


with open(FILEPATH_OUTPUT+F_NAME_STUDENTS, "w") as f_out:
    json.dump(students,f_out)
pk.dump(students,gzip.open(FILEPATH_OUTPUT+F_NAME_STUDENTS,'wb'))
```

Appendix C: Loading Students

```python
import tqdm
import pandas as pd
import tqdm
import pandas as pd
import dill as pk
import gzip
import json
import datetime
from datetime import datetime

FILEPATH_PROBLEM_LEVEL = '../data/PSAHQV-02-01-2021-13-26-42-ProblemLevel.csv'
FILEPATH_OUTPUT = '../export/'
F_NAME_MISTAKE_MESSAGES = 'mistake_messages.p.gzip'
F_NAME_STUDENTS = 'students.json'
PATH =FILEPATH_OUTPUT+F_NAME_STUDENTS

with gzip.open(FILEPATH_OUTPUT+F_NAME_MISTAKE_MESSAGES,'rb') as f:
    mistake_messages = pk.load(f)

df_prob = pd.read_csv(FILEPATH_PROBLEM_LEVEL,low_memory=False)
problem_ids = pd.Series(list(map(lambda x: '"%s"'%(x),
list(mistake_messages.keys()))))


class Student:
    def __init__(self,user_id='',
        problem_ids=[],
        correct_answers={},
        mistakes={},
        n_mistakes={},
        messages={},
        n_messages={},
        starts={},
        ends={},
        total_times={},
        action_orders={},
        action_timestamps={},
        n_hints={},
        hint_timestamps={},
        answer_timestamps={}):

        self.user_id=user_id
        self.problem_ids=problem_ids
        self.correct_answers=correct_answers
        self.mistakes=mistakes
        self.n_mistakes=n_mistakes
```

```python
        self.messages=messages
        self.n_messages = n_messages
        self.starts=starts
        self.ends=ends
        self.total_times=total_times
        self.action_orders=action_orders
        self.action_timestamps=action_timestamps
        self.n_hints=n_hints
        self.hint_timestamps=hint_timestamps
        self.answer_timestamps=answer_timestamps

def DatetimeDecoder(d):
    return datetime.strptime(d, '%m/%d/%Y %H:%M:%S.%f')

def DictDecoder(d,decoder):
    new_d = {}
    for key in d:
        if type(d[key]) == list:
            tmp = []
            for item in d[key]:
                tmp.append(decoder(item))
            new_d[key] = tmp
        else:
            new_d[key] = decoder(d[key])
    return new_d

def decode_student(s):
    return Student(
        user_id=s['user_id'],
        problem_ids=s['problem_ids'],
        correct_answers=s['correct_answers'],
        mistakes=s['mistakes'],
        n_mistakes=s['n_mistakes'],
        messages=s['messages'],
        n_messages=s['n_messages'],
        starts=DictDecoder(s['starts'],DatetimeDecoder),
        ends=DictDecoder(s['ends'],DatetimeDecoder),
        total_times=s['total_times'],
        action_orders=s['action_orders'],
        action_timestamps=DictDecoder(s['action_timestamps'],DatetimeDecoder),
        n_hints=s['n_hints'],
        hint_timestamps=DictDecoder(s['hint_timestamps'],DatetimeDecoder),
        answer_timestamps=DictDecoder(s['answer_timestamps'],DatetimeDecoder)
    )


with open(PATH, "r") as f_in:
    students = json.load(f_in)
```

```python
for student in tqdm.tqdm(students):
    students[student] = decode_student(json.loads(students[student]))
```

Appendix D: Filtering Messages + Generating Plots/Spreadsheet

```
from load_students import *
F_NAME_QUESTIONS_S1 = 'questions.p.gzip'
F_NAME_QUESTIONS_S2 = 'questions_S2.p.gzip'

#multiplication
mistakes_a = [
    lambda x,y,z: (y-z) / x,
    lambda x,y,z: z-y,
    lambda x,y,z: (z-y) * x,
    lambda x,y,z: (z+y) / x,
    lambda x,y,z: (z-y-x),
    lambda x,y,z: (y+z),
    lambda x,y,z: ((z-y) / x)+1,
    lambda x,y,z: ((z-y) / x)-1,
]

#division
mistakes_b = [
    lambda x,y,z: (z-y) * x * (-1),
    lambda x,y,z: x * (y+z),
    lambda x,y,z: (z*x) - y,
    lambda x,y,z: z*x,
    lambda x,y,z: z-y,
    lambda x,y,z: z-y-x,
    lambda x,y,z: z*x+y,
    lambda x,y,z: (z-y)/x,
    lambda x,y,z: (z-y)/x*(-1)
]

#multiplication
mistakes_a_labels = {
    '0a': '(z-y) * x * (-1)',
    '1a': 'x * (y+z)',
    '2a': '(z*x) - y',
    '3a': 'z*x',
    '4a': 'z-y',
    '5a': 'z-y-x',
    '6a': 'z*x+y'
}

#division
mistakes_b_labels = {
    '0b': '(z-y) / x',
    '1b': 'z-y',
    '2b': '(z-y) * x',
    '3b': '(z+y) / x',
```

```
    '4b': '(z-y-x)',
    '5b': '(y+z)',
    '6b': '(z-y) / x+1',
    '7b': '(z-y) / x-1'
}


def mistake_label_mapper(m):
    if 'b' in m:
        return mistakes_b_labels[m]
    elif 'a' in m:
        return mistakes_a_labels[m]
    else:
        return 'other'

import ast, math


#helper for evaluate()
locals =  {key: value for (key,value) in vars(math).items() if key[0] != '_'}
locals.update({"abs": abs, "complex": complex, "min": min, "max": max, "pow":
pow, "round": round})

class Visitor(ast.NodeVisitor):
    def visit(self, node):
       if not isinstance(node, self.whitelist):
           raise ValueError(node)
       return super().visit(node)

    whitelist = (ast.Module, ast.Expr, ast.Load, ast.Expression, ast.Add,
ast.Sub, ast.UnaryOp, ast.Num, ast.BinOp,
            ast.Mult, ast.Div, ast.Pow, ast.BitOr, ast.BitAnd, ast.BitXor,
ast.USub, ast.UAdd, ast.FloorDiv, ast.Mod,
            ast.LShift, ast.RShift, ast.Invert, ast.Call, ast.Name)

#evalutes string expression
def evaluate(expr, locals = {}):
    if any(elem in expr for elem in '\n#') : raise ValueError(expr)
    try:
        node = ast.parse(expr.strip(), mode='eval')
        Visitor().visit(node)
        return eval(compile(node, "<string>", "eval"), {'__builtins__': None},
locals)
    except Exception: raise ValueError(expr)


#gets the index of the first mistake
def get_fmi(s,m):
```

```
    if m == 'mistake':
        for i,pid in enumerate(s.problem_ids):
            if s.n_mistakes[pid] != 0:
                return i

#get's string formula for question
def get_equation(nums,typ):
    if typ == 'multiplication':
        l = list(map(lambda x: str(x), nums))
        return '%s*x + %s = %s' % (l[0],l[1],l[2])
    else:
        l = list(map(lambda x: str(x), nums))
        return 'x/%s + %s = %s' % (l[0],l[1],l[2])

#gets answer to question
def get_answer(nums,typ):
    if typ == 'multiplication':
        l = nums
        return (l[2]-l[1])/l[0]
    else:
        l = nums
        return (l[2]-l[1])*l[0]

#converts string mistake or unsolved mistake into mistake
def parse_mistake(mistake):
    m = mistake.strip('"').replace('[','').replace(']','').replace('
','+').replace('%','').replace('-0','-')
    if m[0] == '(' and m[-1] == ')':
        m = m.strip('()')
    m = m.replace('(','*(')
    if m[0] == '*':
        m = m[1:]
    m = evaluate(m)
    if m == int(m): m = int(m)
    return m

#load questions
with gzip.open(FILEPATH_OUTPUT+F_NAME_QUESTIONS_S1,'rb') as f:
    question_dict_s1 = pk.load(f)

with gzip.open(FILEPATH_OUTPUT+F_NAME_QUESTIONS_S2,'rb') as f:
    question_dict_s2 = pk.load(f)

#messages for each question
s1_messages = {}
s2_messages = {}

#types of each question
```

```
message_types_s1 = {}
message_types_s2 = {}


#store mistake messages S1
for question in tqdm.tqdm(question_dict_s1):
    q = question_dict_s1[question]
    message_types_s1[question] = {}
    messages = {}
    if q[0] == 0:
        for i in range(len(mistakes_a)):
            messages[mistakes_a[i](q[1],q[2],q[3])] = []
            message_types_s1[question][mistakes_a[i](q[1],q[2],q[3])] = i
    else:
        for i in range(len(mistakes_b)):
            messages[mistakes_b[i](q[1],q[2],q[3])] = []
            message_types_s1[question][mistakes_b[i](q[1],q[2],q[3])] = i
    s1_messages[question] = messages

#Store mistake messages S2
for question in question_dict_s2:
    q = question_dict_s2[question]
    message_types_s2[question] = {}
    messages = {}
    if q[0] == 0:
        for i in range(len(mistakes_a)):
            messages[mistakes_a[i](q[1],q[2],q[3])] = []
            message_types_s2[question][mistakes_a[i](q[1],q[2],q[3])] = i
    else:
        for i in range(len(mistakes_b)):
            messages[mistakes_b[i](q[1],q[2],q[3])] = []
            message_types_s2[question][mistakes_b[i](q[1],q[2],q[3])] = i
    s2_messages[question] = messages

s1_ = set(question_dict_s1.values())
s2_ = set(question_dict_s2.values())

A = []
B = []

################################################################
#filtering
#remove students with no mistake
tmp = {}
for sid in students:
    s = students[sid]
    fmi = get_fmi(s,'mistake')
    if fmi == None: continue
```

```
    tmp[sid] = s
students = tmp
print(len(students))

#remove students by time
tmp = {}
counter = 0
for sid in students:
    s = students[sid]
    fmi = get_fmi(s,'mistake')
    fmpid = s.problem_ids[fmi]
    if s.answer_timestamps[fmpid][0].year > 2016:
        counter+=1
        tmp[sid] = s
print(counter)
students = tmp
################################################################


#first mistake problem ids
A_fmpids = []
B_fmpids = []

#emperical mistakes made for given pid
s2_mistakes = {}
s1_mistakes = {}

#attempt counter for all problems
s1_total_attempted = {}
s2_total_attempted = {}

#attempt counter for students' first problems
s1_first_attempted = {}
s2_first_attempted = {}

#mistake counter for all problems
s1_total_mistake_counts = {}
s2_total_mistake_counts = {}

#mistake counter for students' first problems
s1_first_mistake_counts = {}
s2_first_mistake_counts = {}


#get difficulty
difficulty_dict = dict(zip(list(df_difficulty['Problem ID']),
list(df_difficulty.iloc[:,3])))
```

```
s1_first_pids = []
s2_first_pids = []

#iterate through students to get information about problems they've solved
for sid in tqdm.tqdm(students):
    s = students[sid]

    #iterate through problems student has actions for
    for q,pid in enumerate(s.problem_ids):
        #problem was in control group

        if pid.strip('"') in s1_messages:
            s1_first_pids.append(pid.strip('"'))

        if pid.strip('"') in s1_messages:
            if q == 0:
                #first problem that student attempted
                if pid.strip('"') not in s1_first_attempted:
s1_first_attempted[pid.strip('"')] = 1
                else: s1_first_attempted[pid.strip('"')] +=1

            if pid.strip('"') not in s1_mistakes: s1_mistakes[pid.strip('"')] =
[]

            if pid.strip('"') not in s1_total_attempted:
s1_total_attempted[pid.strip('"')] = 1
            else: s1_total_attempted[pid.strip('"')] +=1
            A_fmpids.append(pid)

            #student made at least 1 mistake
            if len(s.mistakes[pid]) > 0:
                if q == 0:
                    #first problem student attempted AND a mistake
                    if pid.strip('"') not in s1_first_mistake_counts:
s1_first_mistake_counts[pid.strip('"')] = 1
                    else: s1_first_mistake_counts[pid.strip('"')] +=1

                #ANY Problem was a mistake for student
                if pid.strip('"') not in s1_total_mistake_counts:
s1_total_mistake_counts[pid.strip('"')] = 1
                else: s1_total_mistake_counts[pid.strip('"')] +=1

                #iterate through student's mistakes
                for mistake in s.mistakes[pid]:
                    if "/0" in mistake: continue
                    m = parse_mistake(mistake)

                    #save as emperical mistake for given pid
```

```
                    s1_mistakes[pid.strip('"')].append(m)

                    #mistake was a CWA
                    if m in list(map(lambda x: eval(str(x)),
s1_messages[pid.strip('"')])):
                        A_fmpids.append(pid.strip('"'))
                    break
        break


#iterate through students in s2
for sid in tqdm.tqdm(students):
    s = students[sid]

    #iterate through problems student has actions for
    for q,pid in enumerate(s.problem_ids):
        #problem was in control group

        if pid.strip('"') in s2_messages:
            s2_first_pids.append(pid.strip('"'))

        if pid.strip('"') in s2_messages:
            if q == 0:
                #first problem that student attempted
                if pid.strip('"') not in s2_first_attempted:
s2_first_attempted[pid.strip('"')] = 1
                else: s2_first_attempted[pid.strip('"')] +=1

            if pid.strip('"') not in s2_mistakes: s2_mistakes[pid.strip('"')] =
[]
            #if pid.strip('"') == 'PRATUWT':continue
            if pid.strip('"') not in s2_total_attempted:
s2_total_attempted[pid.strip('"')] = 1
            else: s2_total_attempted[pid.strip('"')] +=1
            B_fmpids.append(pid)

            #student made at least 1 mistake
            if len(s.mistakes[pid]) > 0:
                if q == 0:
                    #first problem student attempted AND a mistake
                    if pid.strip('"') not in s2_first_mistake_counts:
s2_first_mistake_counts[pid.strip('"')] = 1
                    else: s2_first_mistake_counts[pid.strip('"')] +=1

                #ANY Problem was a mistake for student
                if pid.strip('"') not in s2_total_mistake_counts:
s2_total_mistake_counts[pid.strip('"')] = 1
                else: s2_total_mistake_counts[pid.strip('"')] +=1
```

```
                #iterate through student's mistakes
                for mistake in s.mistakes[pid]:
                    if "/0" in mistake: continue
                    m = parse_mistake(mistake)

                    #save as emperical mistake for given pid
                    s2_mistakes[pid.strip('"')].append(m)

                    #mistake was a CWA
                    if m in list(map(lambda x: eval(str(x)),
s2_messages[pid.strip('"')])):
                        B_fmpids.append(pid.strip('"'))
                    break
        break

s1_diff = Counter([difficulty_dict[pid] for pid in s1_first_pids])
s2_diff = Counter([difficulty_dict[pid] for pid in s2_first_pids])

#top observed all answers
s1_hist_all = {pid:Counter(s1_mistakes[pid]) for pid in s1_mistakes}
s2_hist_all = {pid:Counter(s2_mistakes[pid]) for pid in s2_mistakes}

#top observed selent answers
s1_hist_cwa = {pid:Counter([mistake for mistake in s1_mistakes[pid] if mistake in
list(map(lambda x: eval(str(x)), s1_messages[pid]))]) for pid in s1_mistakes}
s2_hist_cwa = {pid:Counter([mistake for mistake in s2_mistakes[pid] if mistake in
list(map(lambda x: eval(str(x)), s2_messages[pid]))]) for pid in s2_mistakes}

IMAGE_OUTPUT = '../figures/'

for pid in s1_hist_all:
    #l1: all students who mistaked pid, l2: students who mistaked pid on first
problem
    l_cwa = {str(mistake):s1_hist_cwa[pid][mistake] for mistake in
s1_hist_cwa[pid]}
    l_all = {str(mistake):s1_hist_all[pid][mistake] for mistake in
s1_hist_all[pid]}

    if question_dict_s1[pid][0] == 0: qtype = 'multiplication'
    else: qtype = 'division'

    fig, (ax1, ax2,ax3) = plt.subplots(1, 3, figsize=(12, 4))

    #top observed wrong answers
    ax1.set_title('Top WA')
    ax1.bar(l_all.keys(), l_all.values())
    fig.autofmt_xdate(rotation=45)
```

```
    #top common wrong answers
    ax2.set_title('Top Selent WA')
    ax2.bar(l_cwa.keys(), l_cwa.values())
    fig.autofmt_xdate(rotation=45)

    #statistics
    ax3.set_xticks([])
    ax3.set_yticks([])
    ax3.set_xlim([0,1])
    ax3.set_ylim([0,1])

    stats = ['Problem ID: %s'%(pid),
    'Group: S1',
    'Problem Type: %s'%qtype,
    'Equation: %s'%get_equation(question_dict_s1[pid][1:4],qtype),
    'Answer: %d'%get_answer(question_dict_s1[pid][1:4],qtype),
    'A: %s'%question_dict_s1[pid][1],
    'B: %s'%question_dict_s1[pid][2],
    'C: %s'%question_dict_s1[pid][3],
    'N Students Wrong First Problem: %d'%s1_first_mistake_counts[pid],
    'N Students Attempted First Problem: %d'%(s1_first_attempted[pid]),
    'First Prob Accuracy:
%.2f%%'%(100*abs(s1_first_attempted[pid]-s1_first_mistake_counts[pid])/s1_first_a
ttempted[pid]),
    'N Students Wrong Overall: %d'%s1_total_mistake_counts[pid],
    'N Students Attempted Overall: %d'%s1_total_attempted[pid],
    'Problem Overall Accuracy:
%.2f%%'%(100*abs(s1_total_mistake_counts[pid]-s1_total_attempted[pid])/s1_total_a
ttempted[pid]),
    'CWA/TOWA: %.2f%%' % (100*sum(list(l_cwa.values()))/(sum(l_all.values())))]

    for i,stat in enumerate(stats):
        ax3.text(.07,1 - (i+1)*(1/(len(stats)+1)),stat)
    for i,stat in enumerate(stats):
        ax3.text(.07,1 - (i+1)*(1/(len(stats)+1)),stat)
    plt.tight_layout()
    plt.show()
    #plt.savefig(IMAGE_OUTPUT + '_s1_'+pid)


for pid in s2_hist_all:
    l1 = {str(mistake):B_hist[pid][mistake] for mistake in B_hist[pid] if
B_hist[pid][mistake]}# >=
max(2,B_hist[pid].most_common()[len(s2_messages[pid])][1])}
    l2 = {str(mistake):B_hist_actual[pid][mistake] for mistake in
B_hist_actual[pid]}

    if question_dict_s2[pid][0] == 0: qtype = 'multiplication'
```

```
    else: qtype = 'division'

    fig, (ax1, ax2,ax3) = plt.subplots(1, 3, figsize=(12, 4))
    #top observed wrong answers
    ax1.set_title('Top Observed WA')
    ax1.bar(l1.keys(), l1.values())
    fig.autofmt_xdate(rotation=90)

    #top common wrong answers
    ax2.set_title('Common WA')
    ax2.bar(l2.keys(), l2.values())
    fig.autofmt_xdate(rotation=90)
    fig.xticks(rotation=45)
    #show statistics
    ax3.set_xticks([])
    ax3.set_yticks([])
    ax3.set_xlim([0,1])
    ax3.set_ylim([0,1])

    stats = ['Problem ID: %s'%(pid),
    'Group: S2',
    'Problem Type: %s'%qtype,
    'Equation: %s'%get_equation(question_dict_s2[pid][1:4],qtype),
    'Answer: %d'%get_answer(question_dict_s2[pid][1:4],qtype),
    'A: %s'%question_dict_s2[pid][1],
    'B: %s'%question_dict_s2[pid][2],
    'C: %s'%question_dict_s2[pid][3],
    'N Students Wrong First Problem: %d'%s2_first_mistake_counts[pid],
    'N Students Attempted First Problem: %d'%(s2_first_attempted[pid]),
    'First Prob Accuracy:
%.2f%%'%(100*abs(s2_first_attempted[pid]-s2_first_mistake_counts[pid])/s2_first_a
ttempted[pid]),
    'N Students Wrong Overall: %d'%s2_total_mistake_counts[pid],
    'N Students Attempted Overall: %d'%s2_total_attempted[pid],
    'Problem Overall Accuracy:
%.2f%%'%(100*abs(s2_total_mistake_counts[pid]-s2_total_attempted[pid])/s2_total_a
ttempted[pid]),
    'CWA/(CWA + TOWA): %.2f%%' %
(100*sum(list(l2.values()))/(sum(list(l1.values()))+sum(list(l2.values()))))]

    for i,stat in enumerate(stats):
        ax3.text(.07,1 - (i+1)*(1/(len(stats)+1)),stat)

    plt.tight_layout()
    plt.savefig(IMAGE_OUTPUT + '_s2_'+pid)


#export dataframe
```

```python
s1_rows = []
s2_rows = []

colnames = ['Problem ID',
    'Group',
    'Problem Type',
    'Equation',
    'Answer',
    'A',
    'B',
    'C',
    'N Students Wrong First Problem',
    'N Students Attempted First Problem',
    'First Prob Accuracy',
    'N Students Wrong Overall',
    'N Students Attempted Overall',
    'Problem Overall Accuracy',
    'CWA/(CWA + TOWA)'
]

dfs1 = pd.DataFrame(s1_rows, columns = colnames)
dfs2 = pd.DataFrame(s2_rows, columns = colnames)

dfs1.to_csv(IMAGE_OUTPUT +'s1.csv')
dfs2.to_csv(IMAGE_OUTPUT +'s2.csv')
```

Appendix E: Overlapping Messages

```python
import pickle as pk
from collections import Counter
import gzip


FILEPATH_OUTPUT = '../export/'

#multiplication
mistakes_a = [
    lambda x,y,z: (y-z) / x,
    lambda x,y,z: z-y,
    lambda x,y,z: (z-y) * x,
    lambda x,y,z: (z+y) / x,
    lambda x,y,z: (z-y-x),
    lambda x,y,z: (y+z),
    lambda x,y,z: ((z-y) / x)+1,
    lambda x,y,z: ((z-y) / x)-1,
]

#division
```

```python
mistakes_b = [
    lambda x,y,z: (z-y) * x * (-1),
    lambda x,y,z: x * (y+z),
    lambda x,y,z: (z*x) - y,
    lambda x,y,z: z*x,
    lambda x,y,z: z-y,
    lambda x,y,z: z-y-x,
    lambda x,y,z: z*x+y,
    lambda x,y,z: (z-y)/x,
    lambda x,y,z: (z-y)/x*(-1)
]

all_first_mistakes=pk.load(gzip.open(FILEPATH_OUTPUT+"all_mistakes.p.gzip",'rb'))
all_cwa_types=pk.load(gzip.open(FILEPATH_OUTPUT+"all_cwa_types.p.gzip",'rb'))
all_question_dict=pk.load(gzip.open(FILEPATH_OUTPUT+"all_question_dict.p.gzip",'r
b'))


def get_correct_ans(nums,typ):
    if typ == 0:
        eq = lambda  x,y,z: (z-y)/x
    if typ == 1:
        eq = lambda  x,y,z: (z-y)*x

def get_n_repeated(l):
    d = {}
    c = Counter(l)
    acc = []

    for x in set(l):
        acc.append(c[x]) #save counts
    return len(set(acc) - set([1]))

def get_repeated_indeces(l):
    c = dict(Counter(l))
    print(c)
    c = {k:[i for i, v in enumerate(l) if v == k] for k in c if c[k]>1}
    return c

def get_equation(nums,typ):
    if typ == 0:
        l = list(map(lambda x: str(x), nums))
        return '%s*x + %s = %s' % (l[0],l[1],l[2])
    else:
        l = list(map(lambda x: str(x), nums))
        return 'x/%s + %s = %s' % (l[0],l[1],l[2])

#calculate which problems have repeated messages
```

```python
for pid in all_question_dict:
    q = all_question_dict[pid]
    a=q[1]
    b=q[2]
    c=q[3]
    ECWAS = []

    if q[0] == 0:
        for i in range(len(mistakes_a)):
            ECWAS.append(mistakes_a[i](a,b,c))
    else:
        for i in range(len(mistakes_b)):
            ECWAS.append(mistakes_b[i](a,b,c)
    #length less than n ECWAs means we have repeats
    print(pid, len(set(ECWAS)))
```

Appendix F: Problem Spreadsheet

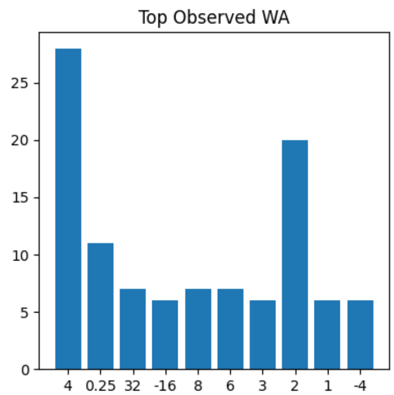
Problem ID	Group		Type	Equation		Answer	A	B	C	Problem Overall Accuracy	First Prob Accuracy	CWA/(TOWA)	B < C	Accuracy Not First	First acc > Not First acc	N Students Wrong First Problem	N Students Attempted First Problem	N Students Wrong Overall	N Students Attempted Overall	Accuracy diff
PRATU5D	S1		division	x/7 + 8 = 1		-49	7	8	1	36.97%	14.29%	43.75%	FALSE	43.83%	TRUE	42	49	133	211	29.54%
PRATU5R	S1		division	x/6 + 11 = 5		-36	6	11	5	44.97%	12.50%	31.03%	FALSE	52.55%	TRUE	28	32	93	169	40.05%
PRATU5H	S1		division	x/7 + 11 = -1		-84	7	11	-1	45.61%	15.00%	41.63%	FALSE	54.96%	TRUE	34	40	93	171	39.96%
PRATU5Z	S1		division	x/11 + 9 = 2		-77	11	9	2	49.44%	18.52%	38.79%	FALSE	54.90%	TRUE	22	27	91	180	36.38%
PRATU5X	S1		division	x/3 + 10 = 2		-24	3	10	2	50.00%	16.67%	43.98%	FALSE	57.25%	TRUE	25	30	84	168	40.58%
PRATU52	S1		division	x/9 + 7 = 3		-36	9	7	3	53.22%	29.41%	41.72%	FALSE	59.12%	TRUE	24	34	80	171	29.71%
PRATU55	S1		division	x/3 + 9 = 4		-15	3	9	4	55.56%	44.44%	41.26%	FALSE	57.78%	TRUE	15	27	72	162	13.34%
PRATU5Q	S1		division	x/7 + 2 = 9		49	7	2	9	56.41%	39.29%	32.35%	TRUE	60.16%	TRUE	17	28	68	156	20.87%
PRATU54	S1		division	x/4 + 4 = 3		-4	4	4	3	57.46%	22.86%	43.27%	FALSE	65.75%	TRUE	27	35	77	181	42.89%
PRATU5E	S1		division	x/5 + 6 = 5		-5	5	6	5	57.96%	31.03%	38.36%	FALSE	64.06%	TRUE	20	29	66	157	33.03%
PRATU48	S1		division	x/6 + 8 = 4		-24	6	8	4	58.16%	17.65%	38.13%	FALSE	63.71%	TRUE	14	17	59	141	46.06%
PRATU5K	S1		division	x/6 + 5 = 3		-12	6	5	3	59.02%	21.21%	35.57%	FALSE	67.33%	TRUE	26	33	75	183	46.12%
PRATU5V	S1		division	x/7 + 5 = 0		-35	7	5	0	61.18%	40.00%	47.45%	FALSE	65.71%	TRUE	18	30	66	170	25.71%
PRATU5Y	S1		division	x/11 + 9 = 7		-22	11	9	7	62.21%	22.73%	41.43%	FALSE	68.00%	TRUE	17	22	65	172	45.27%
PRATU5P	S1		division	x/10 + 5 = 3		-20	10	5	3	62.66%	30.00%	37.88%	FALSE	67.39%	TRUE	14	20	59	158	37.39%
PRATU5F	S1		division	x/8 + 2 = 4		16	8	2	4	63.10%	42.86%	28.77%	TRUE	67.14%	TRUE	16	28	62	168	24.28%
PRATU5T	S1		division	x/2 + 5 = 7		4	2	5	7	63.64%	41.38%	29.55%	TRUE	68.03%	TRUE	17	29	64	176	26.65%
PRATU5A	S1		division	x/8 + 7 = 6		-8	8	7	6	66.46%	44.44%	40.71%	FALSE	70.90%	TRUE	15	27	54	161	26.46%
PRATU5B	S1		division	x/1 + 5 = -1		-6	1	5	-1	68.42%	64.00%	38.02%	FALSE	69.18%	TRUE	9	25	54	171	5.18%

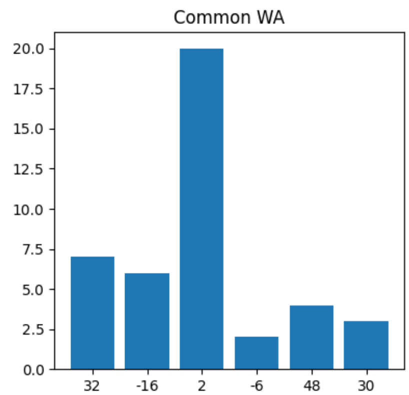
PRATU5W	S1		division	x/1 + 6 = 2		-4	1	6	2	68.65%	54.55%	37.96%	FALSE	73.05%	TRUE	20	44	58	185	18.50%
PRATU53	S1		division	x/1 + 9 = 7		-2	1	9	7	74.52%	55.00%	32.53%	FALSE	77.37%	TRUE	9	20	40	157	22.37%
PRATU5U	S1		division	x/1 + 5 = 4		-1	1	5	4	75.56%	61.76%	43.00%	FALSE	78.77%	TRUE	13	34	44	180	17.01%
PRATU49	S1		division	x/1 + 10 = 11		1	1	10	11	85.16%	72.22%	25.71%	TRUE	86.86%	TRUE	5	18	23	155	14.64%
PRATU5G	S1		division	x/1 + 2 = 7		5	1	2	7	87.26%	79.17%	29.63%	TRUE	88.72%	TRUE	5	24	20	157	9.55%
PRATU5S	S1		division	x/1 + 4 = 6		2	1	4	6	90.00%	84.21%	42.11%	TRUE	90.91%	TRUE	3	19	14	140	6.70%
PRA26BE	S2		division	x/9 + 10 = 5		-45	9	10	5	43.90%	14.29%	45.27%	FALSE	50.00%	TRUE	24	28	92	164	35.71%
PRA26BQ	S2		division	x/6 + 8 = 2		-36	6	8	2	45.09%	29.17%	35.54%	FALSE	47.65%	TRUE	17	24	95	173	18.48%
PRA26BF	S2		division	x/2 + 6 = 4		-4	2	6	4	45.28%	22.58%	37.24%	FALSE	50.78%	TRUE	24	31	87	159	28.20%
PRA26B3	S2		division	x/-1 + 11 = -2		13	-1	11	-2	47.06%	20.00%	43.65%	FALSE	52.86%	TRUE	24	30	90	170	32.86%
PRA26BZ	S2		division	x/-1 + 9 = 0		9	-1	9	0	47.54%	34.48%	44.95%	FALSE	50.00%	TRUE	19	29	96	183	15.52%
PRA26BN	S2		division	x/8 + 6 = 4		-16	8	6	4	48.43%	28.12%	37.28%	FALSE	53.54%	TRUE	23	32	82	159	25.42%
PRA26B2	S2		division	x/4 + 7 = -1		-32	4	7	-1	48.82%	33.33%	33.84%	FALSE	52.14%	TRUE	20	30	87	170	18.81%
PRA26B4	S2		division	x/-1 + 2 = 0		2	-1	2	0	49.10%	25.00%	46.61%	FALSE	54.81%	TRUE	24	32	85	167	29.81%
PRA26BR	S2		division	x/2 + 4 = -1		-10	2	4	-1	50.00%	24.14%	46.89%	FALSE	55.10%	TRUE	22	29	88	176	30.96%
PRA26BV	S2		division	x/7 + 2 = -1		-21	7	2	-1	51.72%	18.18%	43.75%	FALSE	59.57%	TRUE	27	33	84	174	41.39%
PRA26BC	S2		division	x/6 + 9 = 3		-36	6	9	3	52.73%	9.52%	36.00%	FALSE	59.03%	TRUE	19	21	78	165	49.51%
PRA26BU	S2		division	x/10 + 3 = 1		-20	10	3	1	56.36%	22.22%	37.44%	FALSE	63.04%	TRUE	21	27	72	165	40.82%
PRA26B6	S2		division	x/2 + 6 = 1		-10	2	6	1	57.14%	36.67%	43.51%	FALSE	61.83%	TRUE	19	30	69	161	25.16%
PRA26BM	S2		division	x/6 + 4 = 3		-6	6	4	3	58.28%	26.67%	38.93%	FALSE	66.12%	TRUE	22	30	63	151	39.45%
PRA26BK	S2		division	x/10 + 3 = 2		-10	10	3	2	58.44%	30.00%	40.28%	FALSE	65.32%	TRUE	21	30	64	154	35.32%
PRA26BG	S2		division	x/3 + 6 = 9		9	3	6	9	58.50%	26.67%	34.09%	TRUE	64.12%	TRUE	22	30	83	200	37.45%
PRA26BJ	S2		division	x/-2 + 6 = 6		0	-2	6	6	62.65%	52.38%	35.62%	FALSE	64.14%	TRUE	10	21	62	166	11.76%
PRA26BD	S2		division	x/5 + 2 = 5		15	5	2	5	63.12%	42.86%	38.69%	TRUE	66.19%	TRUE	12	21	59	160	23.33%
PRA26BX	S2		division	x/8 + 5 = 7		16	8	5	7	67.48%	35.71%	32.99%	TRUE	74.07%	TRUE	18	28	53	163	38.36%
PRA26BY	S2		division	x/4 + 9 = 9		0	4	9	9	68.83%	51.61%	21.51%	FALSE	73.17%	TRUE	15	31	48	154	21.56%
PRA26BP	S2		division	x/5 + 4 = 7		15	5	4	7	72.33%	35.00%	36.46%	TRUE	77.70%	TRUE	13	20	44	159	42.70%
PRA26BW	S2		division	x/11 + 3 = 5		22	11	3	5	75.97%	47.83%	33.33%	TRUE	80.92%	TRUE	12	23	37	154	33.09%
PRA26BS	S2		division	x/1 + 3 = 4		1	1	3	4	85.51%	84.00%	37.78%	TRUE	85.84%	TRUE	4	25	20	138	1.84%
PRATUW7	S1		multiplication	10*x + 8 = -2		-1	10	8	-2	51.59%	44.00%	46.70%	FALSE	53.03%	TRUE	14	25	76	157	9.03%
PRATUW3	S1		multiplication	11*x + 5 = -6		-1	11	5	-6	51.95%	29.63%	44.79%	FALSE	56.69%	TRUE	19	27	74	154	27.06%
PRATUWZ	S1		multiplication	10*x + 3 = -27		-3	10	3	-27	52.35%	40.00%	42.26%	FALSE	54.84%	TRUE	15	25	71	149	14.84%
PRATUW5	S1		multiplication	2*x + 11 = 3		-4	2	11	3	54.82%	23.08%	41.00%	FALSE	60.71%	TRUE	20	26	75	166	37.63%
PRATUWT	S1		multiplication	11*x + 8 = -25		-3	11	8	-25	58.39%	37.50%	39.76%	FALSE	63.57%	TRUE	20	32	67	161	26.07%

PRATUWV	S1		multiplication	7*x + 2 = -26		-4	7	2	-26	59.35%	48.00%	41.13%	FALSE	61.54%	TRUE	13	25	63	155	13.54%
PRATUXF	S1		multiplication	6*x + 10 = -2		-2	6	10	-2	59.44%	50.00%	34.01%	FALSE	60.98%	TRUE	10	20	58	143	10.98%
PRATUW9	S1		multiplication	4*x + 5 = -11		-4	4	5	-11	63.19%	34.78%	34.59%	FALSE	67.86%	TRUE	15	23	60	163	33.08%
PRATUXH	S1		multiplication	9*x + 8 = 8		0	9	8	8	71.33%	58.82%	32.53%	FALSE	72.93%	TRUE	7	17	43	150	14.11%
PRATUXB	S1		multiplication	10*x + 9 = 9		0	10	9	9	74.23%	53.57%	33.33%	FALSE	78.52%	TRUE	13	28	42	163	24.95%
PRATUWS	S1		multiplication	2*x + 4 = 18		7	2	4	18	75.51%	54.55%	44.44%	TRUE	79.20%	TRUE	10	22	36	147	24.65%
PRATUW4	S1		multiplication	2*x + 7 = 15		4	2	7	15	75.52%	60.71%	42.11%	TRUE	79.13%	TRUE	11	28	35	143	18.42%
PRATUXD	S1		multiplication	4*x + 10 = 10		0	4	10	10	78.44%	61.90%	37.50%	FALSE	80.82%	TRUE	8	21	36	167	18.92%
PRATUWX	S1		multiplication	6*x + 8 = 14		1	6	8	14	78.70%	64.00%	45.57%	TRUE	81.25%	TRUE	9	25	36	169	17.25%
PRATUXJ	S1		multiplication	4*x + 11 = 39		7	4	11	39	80.12%	68.00%	38.33%	TRUE	82.27%	TRUE	8	25	33	166	14.27%
PRATUW2	S1		multiplication	7*x + 6 = 41		5	7	6	41	82.39%	80.77%	39.47%	TRUE	82.71%	TRUE	5	26	28	159	1.94%
PRATUXA	S1		multiplication	6*x + 7 = 31		4	6	7	31	82.91%	75.00%	37.50%	TRUE	83.80%	TRUE	4	16	27	158	8.80%
PRATUXE	S1		multiplication	5*x + 10 = 45		7	5	10	45	83.23%	57.89%	38.00%	TRUE	86.62%	TRUE	8	19	27	161	28.73%
PRATUW8	S1		multiplication	6*x + 11 = 23		2	6	11	23	84.62%	68.18%	44.00%	TRUE	87.60%	TRUE	7	22	22	143	19.42%
PRATUWU	S1		multiplication	7*x + 2 = 51		7	7	2	51	84.81%	86.96%	42.59%	TRUE	84.44%	FALSE	3	23	24	158	-2.52%
PRATUW6	S1		multiplication	6*x + 6 = 18		2	6	6	18	84.97%	59.09%	40.79%	TRUE	89.31%	TRUE	9	22	23	153	30.22%
PRATUWY	S1		multiplication	11*x + 7 = 51		4	11	7	51	85.38%	69.23%	44.44%	TRUE	88.28%	TRUE	8	26	25	171	19.05%
PRATUWW	S1		multiplication	5*x + 2 = 42		8	5	2	42	87.25%	79.31%	40.00%	TRUE	89.17%	TRUE	6	29	19	149	9.86%
PRATUXC	S1		multiplication	9*x + 7 = 52		5	9	7	52	87.34%	70.83%	38.71%	TRUE	90.30%	TRUE	7	24	20	158	19.47%
PRATUXG	S1		multiplication	10*x + 6 = 76		7	10	6	76	87.59%	68.42%	44.44%	TRUE	90.68%	TRUE	6	19	17	137	22.26%
PRA26AJ	S2		multiplication	6*x + 6 = -6		-2	6	6	-6	52.00%	17.65%	42.15%	FALSE	60.28%	TRUE	28	34	84	175	42.63%
PRA26AE	S2		multiplication	3*x + 3 = -3		-2	3	3	-3	52.17%	21.21%	42.31%	FALSE	58.94%	TRUE	26	33	88	184	37.73%
PRA2594	S2		multiplication	4*x + 10 = -6		-4	4	10	-6	58.58%	44.44%	42.61%	FALSE	61.27%	TRUE	15	27	70	169	16.83%
PRA26AC	S2		multiplication	4*x + 8 = -8		-4	4	8	-8	59.06%	52.63%	41.67%	FALSE	59.87%	TRUE	9	19	70	171	7.24%
PRA259U	S2		multiplication	7*x + 2 = -19		-3	7	2	-19	66.90%	47.83%	39.84%	FALSE	70.49%	TRUE	12	23	48	145	22.66%
PRA2596	S2		multiplication	6*x + 2 = -22		-4	6	2	-22	68.45%	39.29%	38.41%	FALSE	74.29%	TRUE	17	28	53	168	35.00%
PRA2592	S2		multiplication	9*x + 6 = 6		0	9	6	6	72.88%	51.72%	22.33%	FALSE	77.03%	TRUE	14	29	48	177	25.31%
PRA259Y	S2		multiplication	7*x + 6 = 6		0	7	6	6	73.30%	71.43%	33.64%	FALSE	73.55%	TRUE	6	21	47	176	2.12%
PRA259W	S2		multiplication	6*x + 5 = 5		0	6	5	5	74.58%	53.85%	26.67%	FALSE	78.15%	TRUE	12	26	45	177	24.30%
PRA26AG	S2		multiplication	4*x + 5 = 13		2	4	5	13	79.59%	73.08%	43.64%	TRUE	80.99%	TRUE	7	26	30	147	7.91%
PRA2595	S2		multiplication	11*x + 9 = 31		2	11	9	31	79.88%	63.64%	41.18%	TRUE	82.39%	TRUE	8	22	33	164	18.75%
PRA26AB	S2		multiplication	9*x + 7 = 61		6	9	7	61	81.10%	62.50%	40.91%	TRUE	84.29%	TRUE	9	24	31	164	21.79%
PRA259Z	S2		multiplication	9*x + 3 = 39		4	9	3	39	81.76%	52.63%	43.48%	TRUE	85.71%	TRUE	9	19	29	159	33.08%
PRA259S	S2		multiplication	8*x + 2 = 18		2	8	2	18	81.82%	61.76%	38.78%	TRUE	87.02%	TRUE	13	34	30	165	25.26%

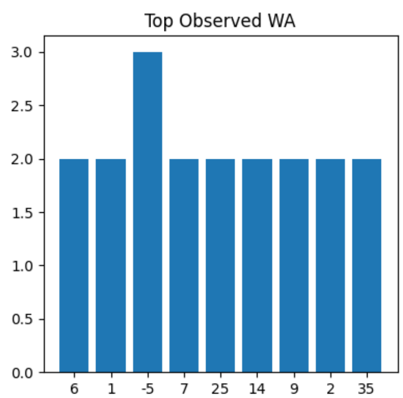
PRA26AF	S2		multiplication	6*x + 10 = 52		7	6	10	52	82.32%	50.00%	42.65%	TRUE	88.41%	TRUE	13	26	29	164	38.41%
PRA259R	S2		multiplication	9*x + 10 = 28		2	9	10	28	82.35%	72.73%	41.67%	TRUE	83.97%	TRUE	6	22	27	153	11.24%
PRA2597	S2		multiplication	4*x + 11 = 31		5	4	11	31	84.28%	66.67%	35.00%	TRUE	87.41%	TRUE	8	24	25	159	20.74%
PRA26AA	S2		multiplication	10*x + 7 = 27		2	10	7	27	84.56%	59.26%	43.08%	TRUE	90.16%	TRUE	11	27	23	149	30.90%
PRA2599	S2		multiplication	7*x + 5 = 61		8	7	5	61	84.66%	87.80%	40.00%	TRUE	83.78%	FALSE	5	41	29	189	-4.02%
PRA259T	S2		multiplication	9*x + 4 = 67		7	9	4	67	85.28%	84.00%	44.44%	TRUE	85.51%	TRUE	4	25	24	163	1.51%
PRA2593	S2		multiplication	7*x + 5 = 19		2	7	5	19	86.18%	81.82%	41.30%	TRUE	86.52%	TRUE	2	11	21	152	4.70%
PRA26AD	S2		multiplication	9*x + 11 = 65		6	9	11	65	86.59%	81.25%	42.37%	TRUE	87.16%	TRUE	3	16	22	164	5.91%
PRA259X	S2		multiplication	9*x + 4 = 76		8	9	4	76	88.19%	73.91%	38.46%	TRUE	90.91%	TRUE	6	23	17	144	17.00%
PRA26AH	S2		multiplication	10*x + 7 = 67		6	10	7	67	89.29%	86.96%	39.02%	TRUE	89.74%	TRUE	3	23	15	140	2.78%
PRA259V	S2		multiplication	10*x + 2 = 82		8	10	2	82	90.17%	73.91%	34.15%	TRUE	92.67%	TRUE	6	23	17	173	18.76%

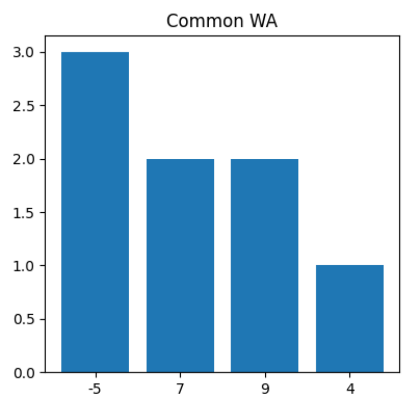
Appendix G: Problem Figures



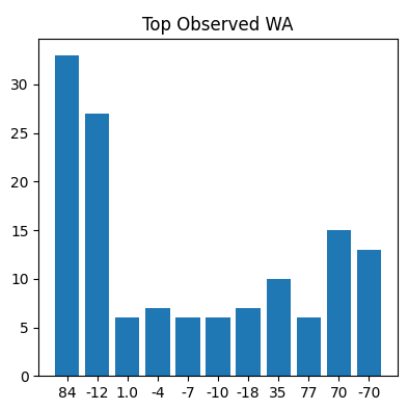


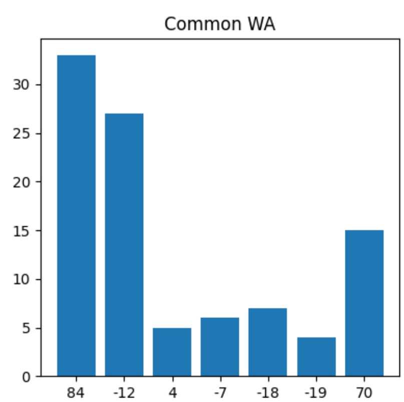
Problem ID: PRATU5F
Group: S1
Problem Type: division
Equation: x/8 + 2 = 4
Answer: 16
A: 8
B: 2
C: 4
N Students Wrong First Problem: 16
N Students Attempted First Problem: 28
First Prob Accuracy: 42.86%
N Students Wrong Overall: 62
N Students Attempted Overall: 168
Problem Overall Accuracy: 63.10%
CWA/(CWA + TOWA): 28.77%



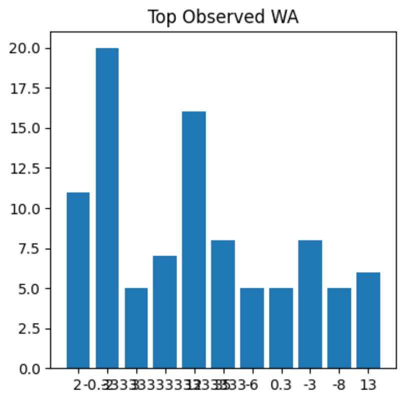


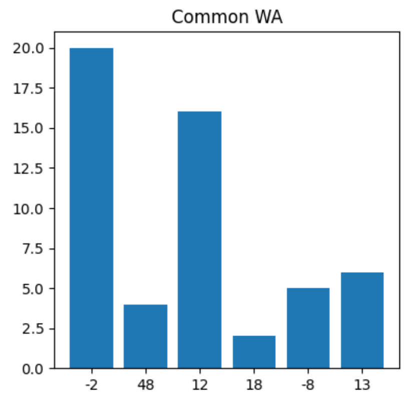
Problem ID: PRATU5G
Group: S1
Problem Type: division
Equation: x/1 + 2 = 7
Answer: 5
A: 1
B: 2
C: 7
N Students Wrong First Problem: 5
N Students Attempted First Problem: 24
First Prob Accuracy: 79.17%
N Students Wrong Overall: 20
N Students Attempted Overall: 157
Problem Overall Accuracy: 87.26%
CWA/(CWA + TOWA): 29.63%



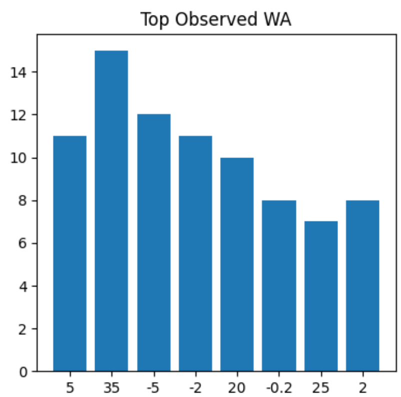


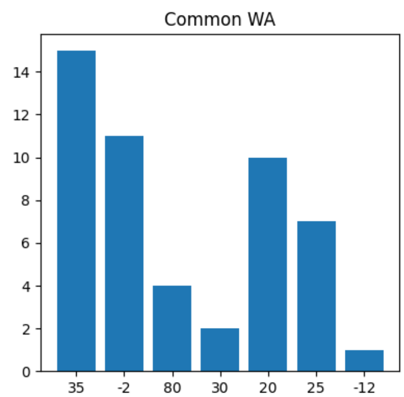
Problem ID: PRATU5H
Group: S1
Problem Type: division
Equation: x/7 + 11 = -1
Answer: -84
A: 7
B: 11
C: -1
N Students Wrong First Problem: 34
N Students Attempted First Problem: 40
First Prob Accuracy: 15.00%
N Students Wrong Overall: 93
N Students Attempted Overall: 171
Problem Overall Accuracy: 45.61%
CWA/(CWA + TOWA): 41.63%



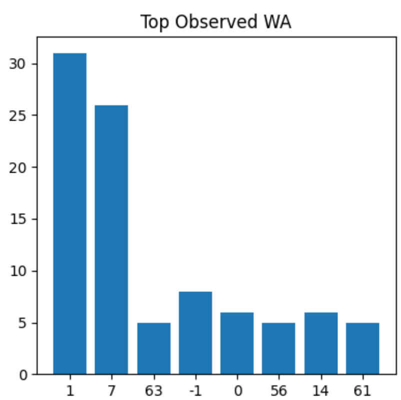


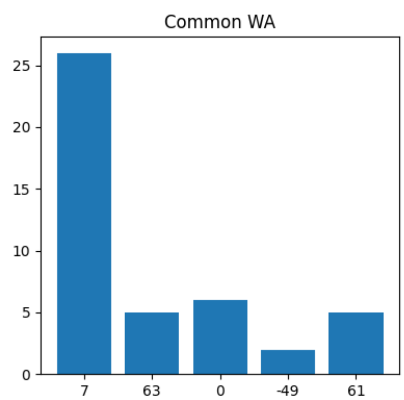
Problem ID: PRATU5K
Group: S1
Problem Type: division
Equation: x/6 + 5 = 3
Answer: -12
A: 6
B: 5
C: 3
N Students Wrong First Problem: 26
N Students Attempted First Problem: 33
First Prob Accuracy: 21.21%
N Students Wrong Overall: 75
N Students Attempted Overall: 183
Problem Overall Accuracy: 59.02%
CWA/(CWA + TOWA): 35.57%



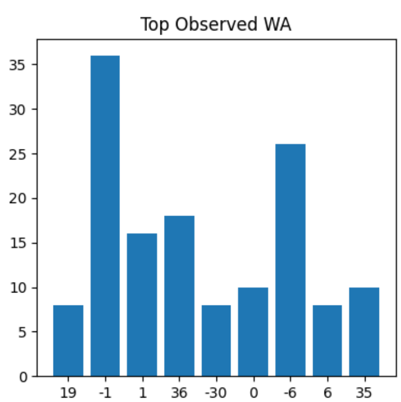


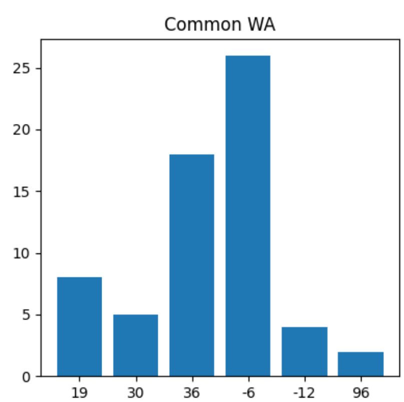
Problem ID: PRATU5P
Group: S1
Problem Type: division
Equation: x/10 + 5 = 3
Answer: -20
A: 10
B: 5
C: 3
N Students Wrong First Problem: 14
N Students Attempted First Problem: 20
First Prob Accuracy: 30.00%
N Students Wrong Overall: 59
N Students Attempted Overall: 158
Problem Overall Accuracy: 62.66%
CWA/(CWA + TOWA): 37.88%



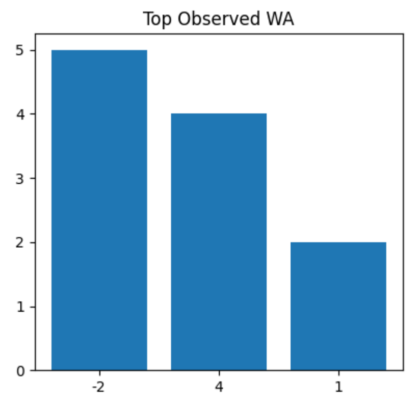


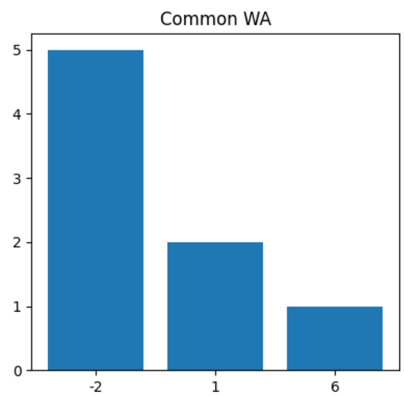
Problem ID: PRATU5Q
Group: S1
Problem Type: division
Equation: x/7 + 2 = 9
Answer: 49
A: 7
B: 2
C: 9
N Students Wrong First Problem: 17
N Students Attempted First Problem: 28
First Prob Accuracy: 39.29%
N Students Wrong Overall: 68
N Students Attempted Overall: 156
Problem Overall Accuracy: 56.41%
CWA/(CWA + TOWA): 32.35%



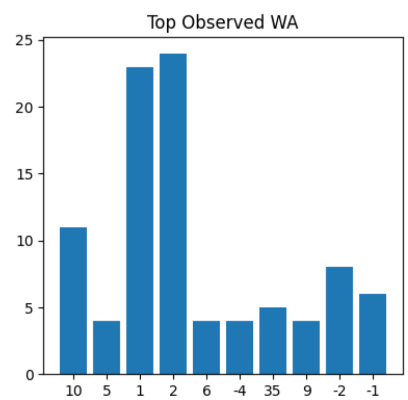


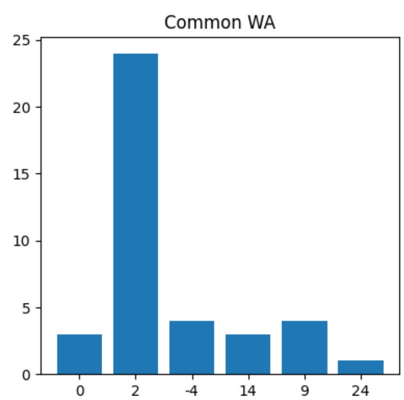
Problem ID: PRATU5R
Group: S1
Problem Type: division
Equation: x/6 + 11 = 5
Answer: -36
A: 6
B: 11
C: 5
N Students Wrong First Problem: 28
N Students Attempted First Problem: 32
First Prob Accuracy: 12.50%
N Students Wrong Overall: 93
N Students Attempted Overall: 169
Problem Overall Accuracy: 44.97%
CWA/(CWA + TOWA): 31.03%



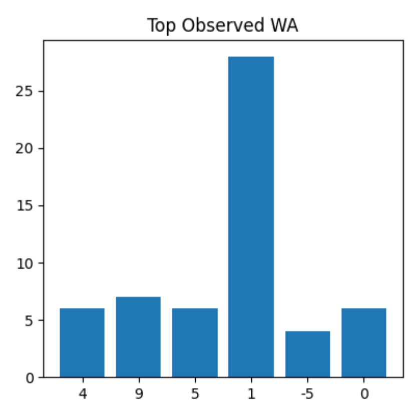


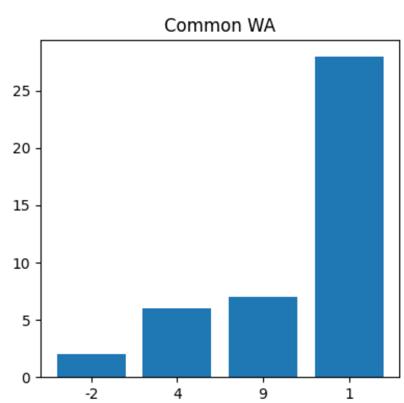
Problem ID: PRATU5S
Group: S1
Problem Type: division
Equation: x/1 + 4 = 6
Answer: 2
A: 1
B: 4
C: 6
N Students Wrong First Problem: 3
N Students Attempted First Problem: 19
First Prob Accuracy: 84.21%
N Students Wrong Overall: 14
N Students Attempted Overall: 140
Problem Overall Accuracy: 90.00%
CWA/(CWA + TOWA): 42.11%



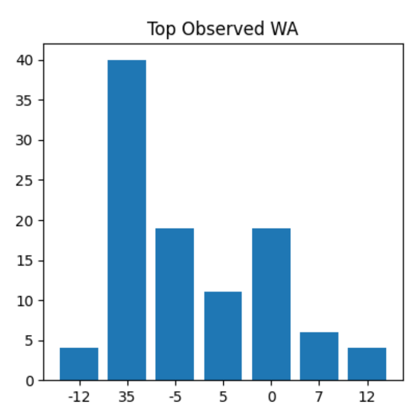


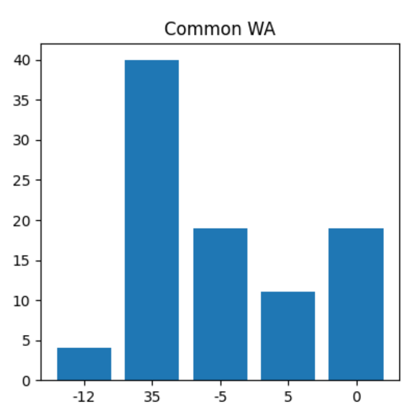
Problem ID: PRATU5T
Group: S1
Problem Type: division
Equation: x/2 + 5 = 7
Answer: 4
A: 2
B: 5
C: 7
N Students Wrong First Problem: 17
N Students Attempted First Problem: 29
First Prob Accuracy: 41.38%
N Students Wrong Overall: 64
N Students Attempted Overall: 176
Problem Overall Accuracy: 63.64%
CWA/(CWA + TOWA): 29.55%



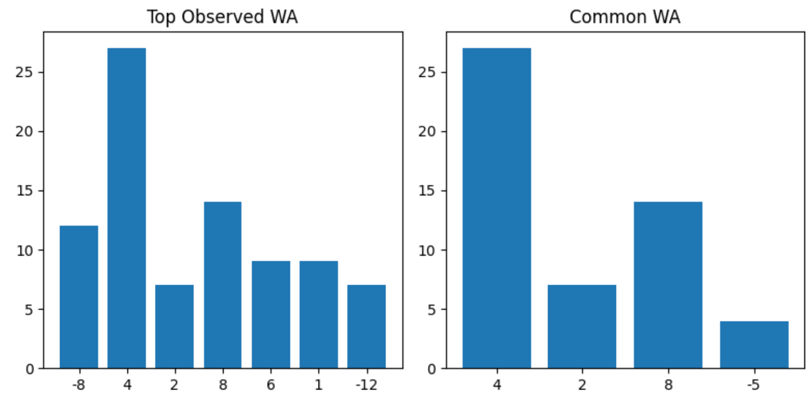


Problem ID: PRATU5U
Group: S1
Problem Type: division
Equation: x/1 + 5 = 4
Answer: -1
A: 1
B: 5
C: 4
N Students Wrong First Problem: 13
N Students Attempted First Problem: 34
First Prob Accuracy: 61.76%
N Students Wrong Overall: 44
N Students Attempted Overall: 180
Problem Overall Accuracy: 75.56%
CWA/(CWA + TOWA): 43.00%

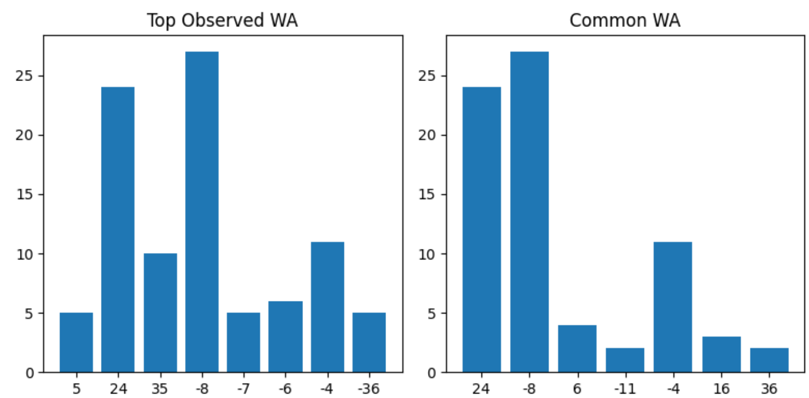




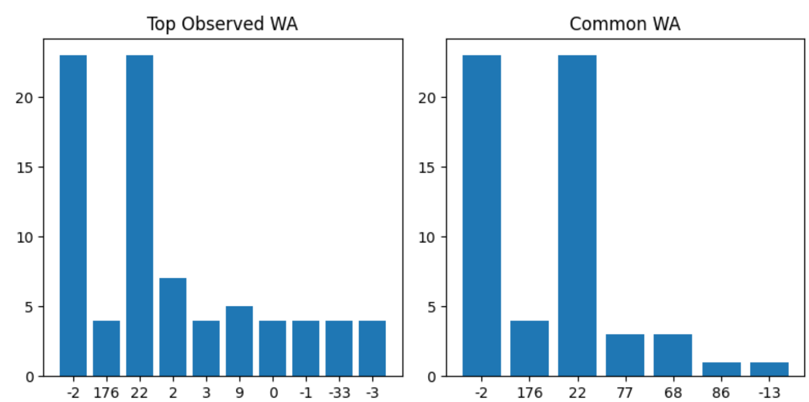
Problem ID: PRATU5V
Group: S1
Problem Type: division
Equation: x/7 + 5 = 0
Answer: -35
A: 7
B: 5
C: 0
N Students Wrong First Problem: 18
N Students Attempted First Problem: 30
First Prob Accuracy: 40.00%
N Students Wrong Overall: 66
N Students Attempted Overall: 170
Problem Overall Accuracy: 61.18%
CWA/(CWA + TOWA): 47.45%



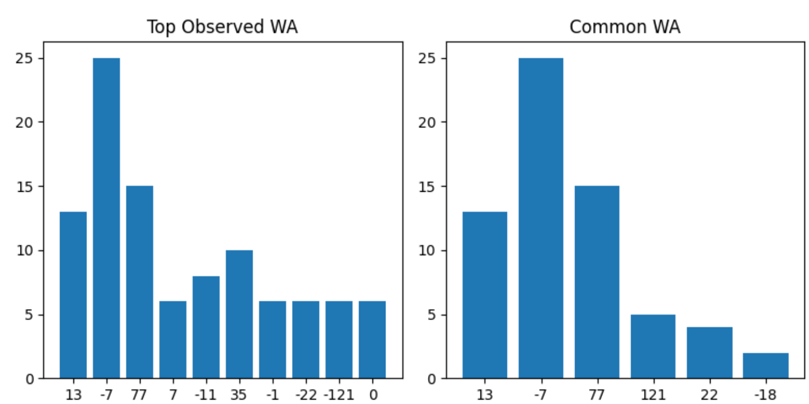
Problem ID: PRATU5W
Group: S1
Problem Type: division
Equation: x/1 + 6 = 2
Answer: -4
A: 1
B: 6
C: 2
N Students Wrong First Problem: 20
N Students Attempted First Problem: 44
First Prob Accuracy: 54.55%
N Students Wrong Overall: 58
N Students Attempted Overall: 185
Problem Overall Accuracy: 68.65%
CWA/(CWA + TOWA): 37.96%



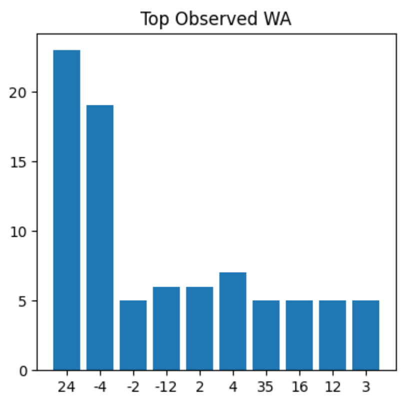
Problem ID: PRATU5X
Group: S1
Problem Type: division
Equation: x/3 + 10 = 2
Answer: -24
A: 3
B: 10
C: 2
N Students Wrong First Problem: 25
N Students Attempted First Problem: 30
First Prob Accuracy: 16.67%
N Students Wrong Overall: 84
N Students Attempted Overall: 168
Problem Overall Accuracy: 50.00%
CWA/(CWA + TOWA): 43.98%

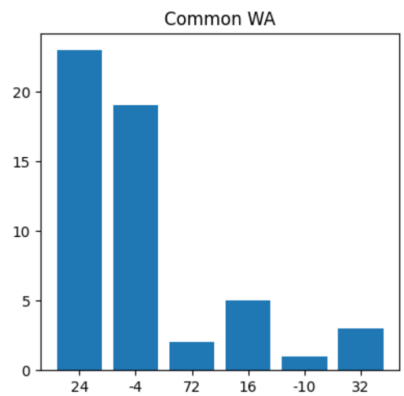


Problem ID: PRATU5Y
Group: S1
Problem Type: division
Equation: x/11 + 9 = 7
Answer: -22
A: 11
B: 9
C: 7
N Students Wrong First Problem: 17
N Students Attempted First Problem: 22
First Prob Accuracy: 22.73%
N Students Wrong Overall: 65
N Students Attempted Overall: 172
Problem Overall Accuracy: 62.21%
CWA/(CWA + TOWA): 41.43%

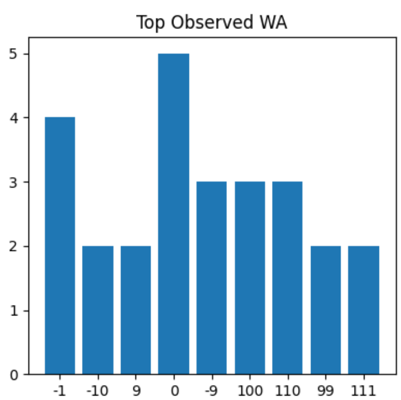


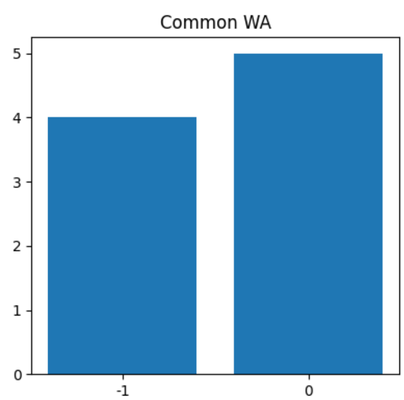
Problem ID: PRATU5Z
Group: S1
Problem Type: division
Equation: x/11 + 9 = 2
Answer: -77
A: 11
B: 9
C: 2
N Students Wrong First Problem: 22
N Students Attempted First Problem: 27
First Prob Accuracy: 18.52%
N Students Wrong Overall: 91
N Students Attempted Overall: 180
Problem Overall Accuracy: 49.44%
CWA/(CWA + TOWA): 38.79%



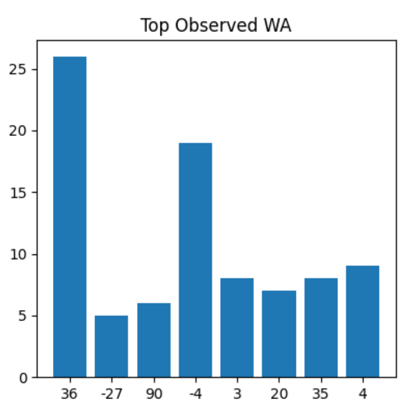


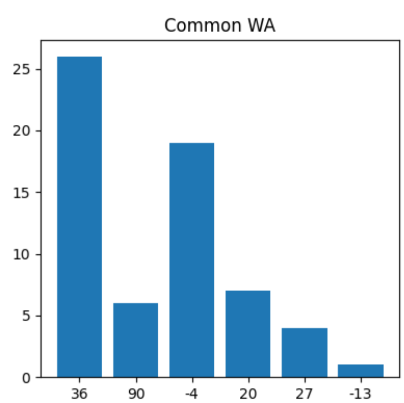
Problem ID: PRATU48
Group: S1
Problem Type: division
Equation: x/6 + 8 = 4
Answer: -24
A: 6
B: 8
C: 4
N Students Wrong First Problem: 14
N Students Attempted First Problem: 17
First Prob Accuracy: 17.65%
N Students Wrong Overall: 59
N Students Attempted Overall: 141
Problem Overall Accuracy: 58.16%
CWA/(CWA + TOWA): 38.13%



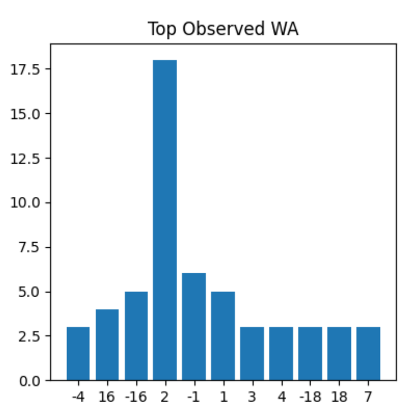


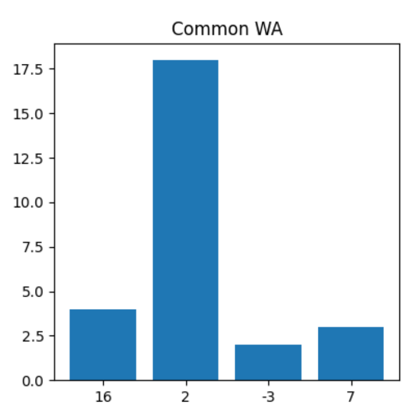
Problem ID: PRATU49
Group: S1
Problem Type: division
Equation: x/1 + 10 = 11
Answer: 1
A: 1
B: 10
C: 11
N Students Wrong First Problem: 5
N Students Attempted First Problem: 18
First Prob Accuracy: 72.22%
N Students Wrong Overall: 23
N Students Attempted Overall: 155
Problem Overall Accuracy: 85.16%
CWA/(CWA + TOWA): 25.71%



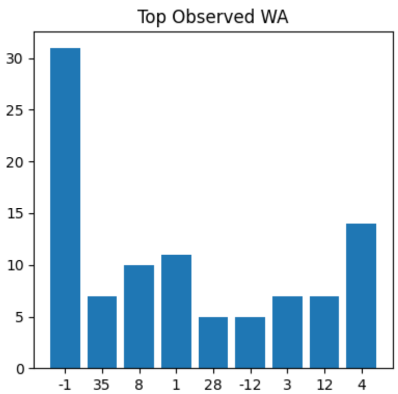


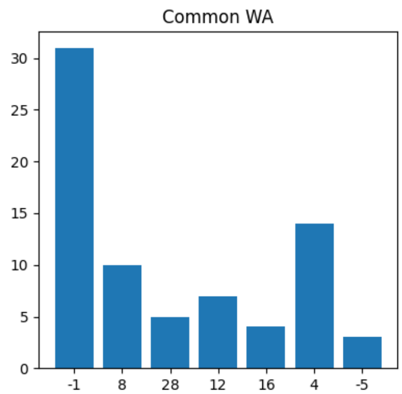
Problem ID: PRATU52
Group: S1
Problem Type: division
Equation: x/9 + 7 = 3
Answer: -36
A: 9
B: 7
C: 3
N Students Wrong First Problem: 24
N Students Attempted First Problem: 34
First Prob Accuracy: 29.41%
N Students Wrong Overall: 80
N Students Attempted Overall: 171
Problem Overall Accuracy: 53.22%
CWA/(CWA + TOWA): 41.72%



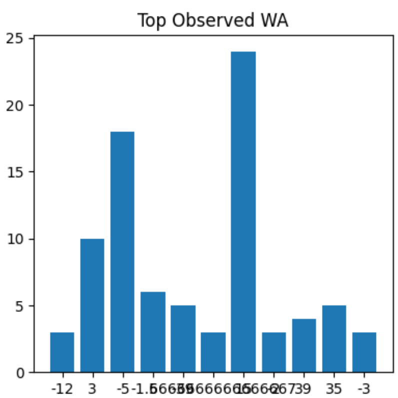


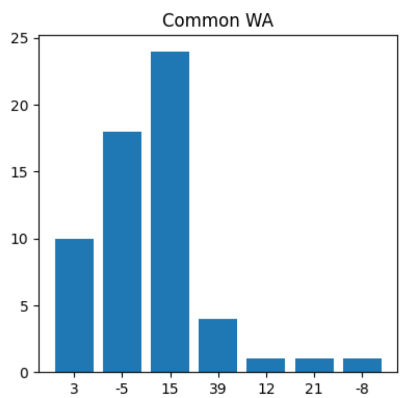
Problem ID: PRATU53
Group: S1
Problem Type: division
Equation: x/1 + 9 = 7
Answer: -2
A: 1
B: 9
C: 7
N Students Wrong First Problem: 9
N Students Attempted First Problem: 20
First Prob Accuracy: 55.00%
N Students Wrong Overall: 40
N Students Attempted Overall: 157
Problem Overall Accuracy: 74.52%
CWA/(CWA + TOWA): 32.53%



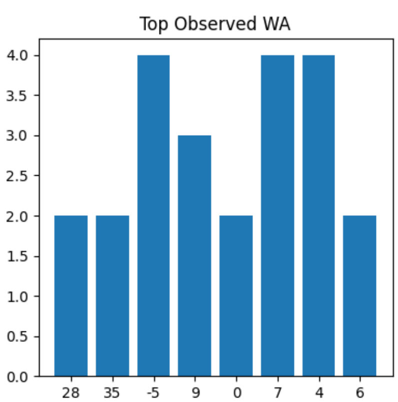


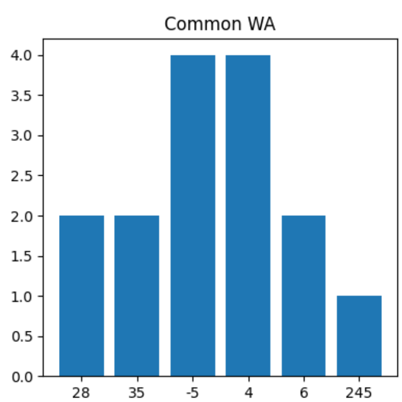
Problem ID: PRATU54
Group: S1
Problem Type: division
Equation: x/4 + 4 = 3
Answer: -4
A: 4
B: 4
C: 3
N Students Wrong First Problem: 27
N Students Attempted First Problem: 35
First Prob Accuracy: 22.86%
N Students Wrong Overall: 77
N Students Attempted Overall: 181
Problem Overall Accuracy: 57.46%
CWA/(CWA + TOWA): 43.27%



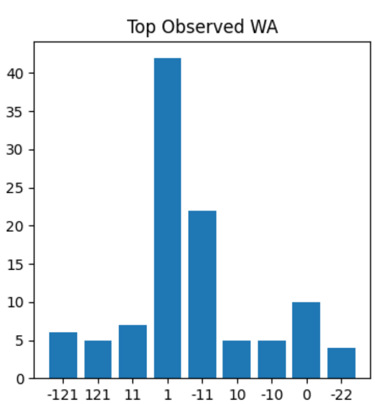


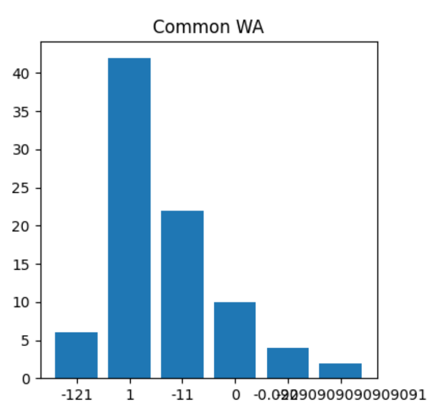
Problem ID: PRATU55
Group: S1
Problem Type: division
Equation: x/3 + 9 = 4
Answer: -15
A: 3
B: 9
C: 4
N Students Wrong First Problem: 15
N Students Attempted First Problem: 27
First Prob Accuracy: 44.44%
N Students Wrong Overall: 72
N Students Attempted Overall: 162
Problem Overall Accuracy: 55.56%
CWA/(CWA + TOWA): 41.26%



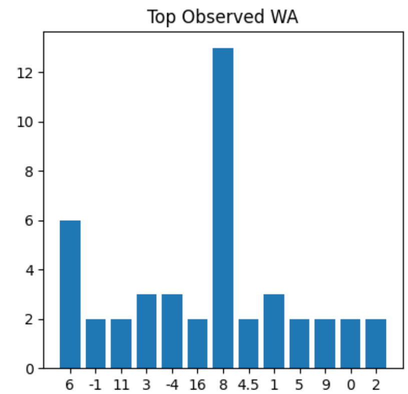


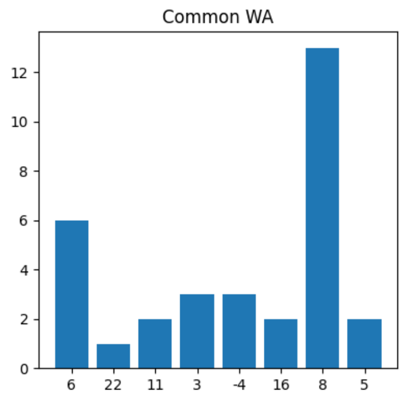
Problem ID: PRATUW2
Group: S1
Problem Type: multiplication
Equation: 7*x + 6 = 41
Answer: 5
A: 7
B: 6
C: 41
N Students Wrong First Problem: 5
N Students Attempted First Problem: 26
First Prob Accuracy: 80.77%
N Students Wrong Overall: 28
N Students Attempted Overall: 159
Problem Overall Accuracy: 82.39%
CWA/(CWA + TOWA): 39.47%



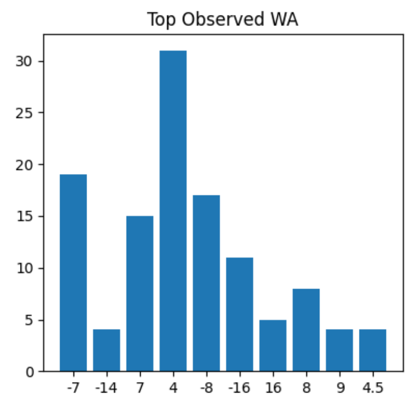


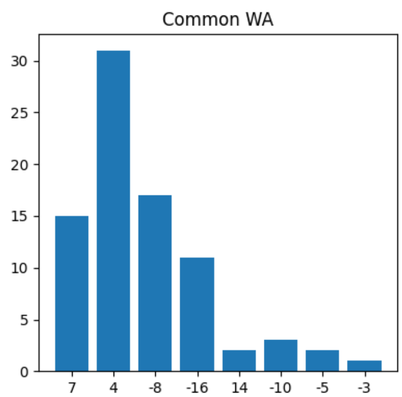
Problem ID: PRATUW3
Group: S1
Problem Type: multiplication
Equation: 11*x + 5 = -6
Answer: -1
A: 11
B: 5
C: -6
N Students Wrong First Problem: 19
N Students Attempted First Problem: 27
First Prob Accuracy: 29.63%
N Students Wrong Overall: 74
N Students Attempted Overall: 154
Problem Overall Accuracy: 51.95%
CWA/(CWA + TOWA): 44.79%



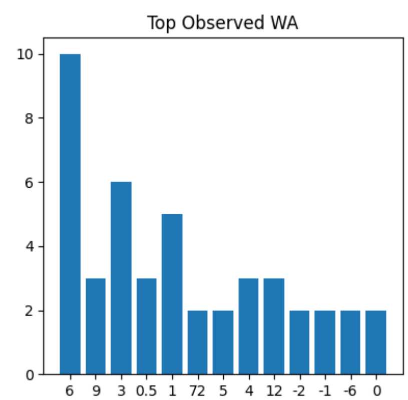


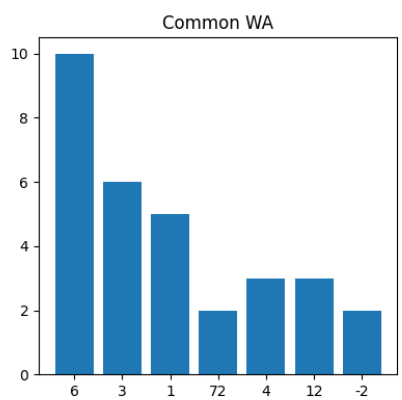
Problem ID: PRATUW4
Group: S1
Problem Type: multiplication
Equation: 2*x + 7 = 15
Answer: 4
A: 2
B: 7
C: 15
N Students Wrong First Problem: 11
N Students Attempted First Problem: 28
First Prob Accuracy: 60.71%
N Students Wrong Overall: 35
N Students Attempted Overall: 143
Problem Overall Accuracy: 75.52%
CWA/(CWA + TOWA): 42.11%



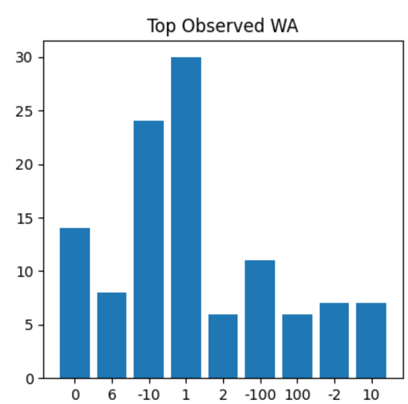


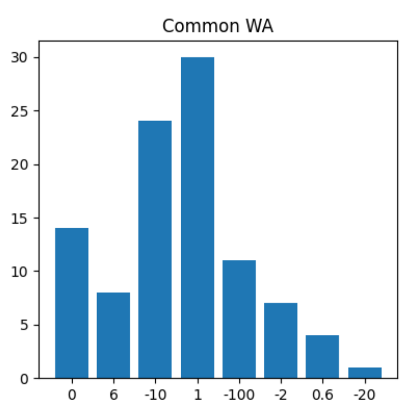
Problem ID: PRATUW5
Group: S1
Problem Type: multiplication
Equation: 2*x + 11 = 3
Answer: -4
A: 2
B: 11
C: 3
N Students Wrong First Problem: 20
N Students Attempted First Problem: 26
First Prob Accuracy: 23.08%
N Students Wrong Overall: 75
N Students Attempted Overall: 166
Problem Overall Accuracy: 54.82%
CWA/(CWA + TOWA): 41.00%



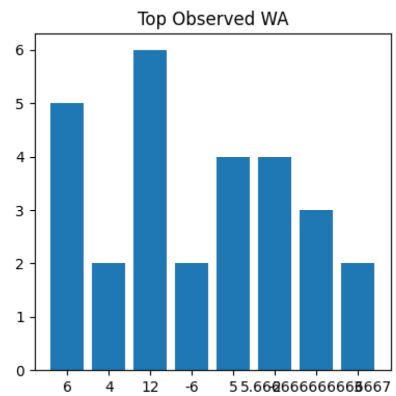


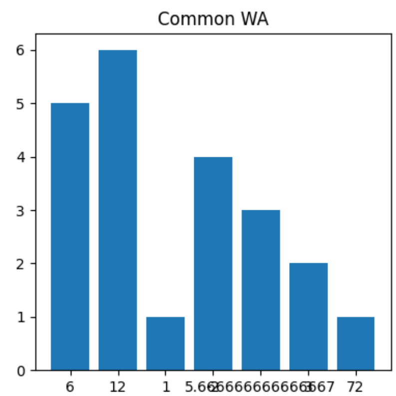
Problem ID: PRATUW6
Group: S1
Problem Type: multiplication
Equation: 6*x + 6 = 18
Answer: 2
A: 6
B: 6
C: 18
N Students Wrong First Problem: 9
N Students Attempted First Problem: 22
First Prob Accuracy: 59.09%
N Students Wrong Overall: 23
N Students Attempted Overall: 153
Problem Overall Accuracy: 84.97%
CWA/(CWA + TOWA): 40.79%



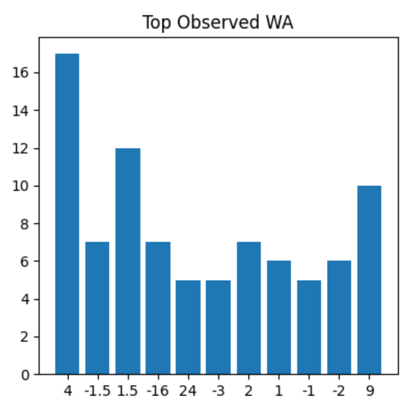


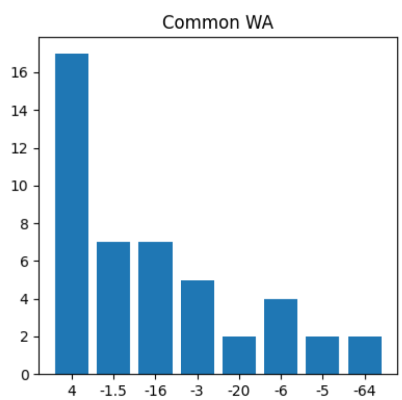
Problem ID: PRATUW7
Group: S1
Problem Type: multiplication
Equation: 10*x + 8 = -2
Answer: -1
A: 10
B: 8
C: -2
N Students Wrong First Problem: 14
N Students Attempted First Problem: 25
First Prob Accuracy: 44.00%
N Students Wrong Overall: 76
N Students Attempted Overall: 157
Problem Overall Accuracy: 51.59%
CWA/(CWA + TOWA): 46.70%



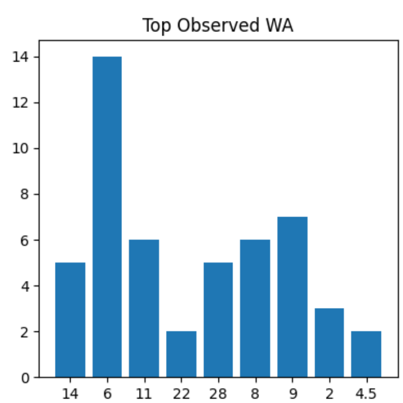


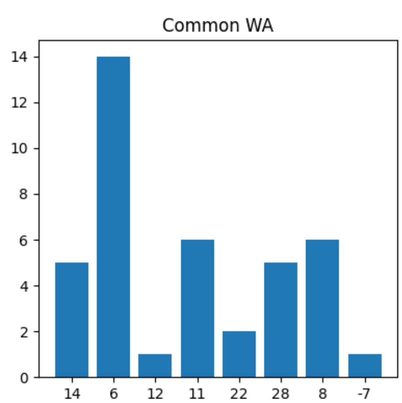
Problem ID: PRATUW8
Group: S1
Problem Type: multiplication
Equation: 6*x + 11 = 23
Answer: 2
A: 6
B: 11
C: 23
N Students Wrong First Problem: 7
N Students Attempted First Problem: 22
First Prob Accuracy: 68.18%
N Students Wrong Overall: 22
N Students Attempted Overall: 143
Problem Overall Accuracy: 84.62%
CWA/(CWA + TOWA): 44.00%



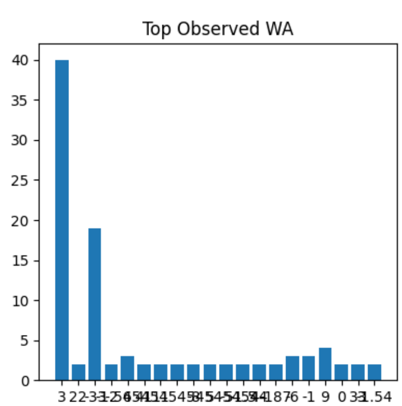


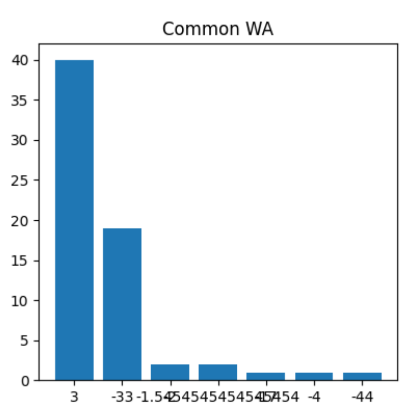
Problem ID: PRATUW9
Group: S1
Problem Type: multiplication
Equation: 4*x + 5 = -11
Answer: -4
A: 4
B: 5
C: -11
N Students Wrong First Problem: 15
N Students Attempted First Problem: 23
First Prob Accuracy: 34.78%
N Students Wrong Overall: 60
N Students Attempted Overall: 163
Problem Overall Accuracy: 63.19%
CWA/(CWA + TOWA): 34.59%



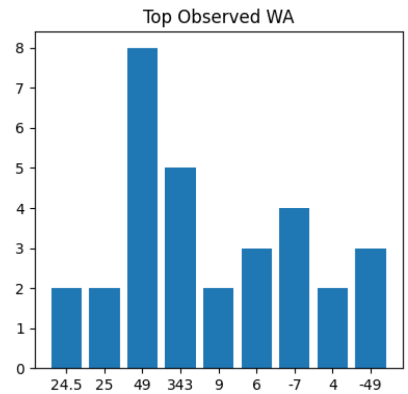


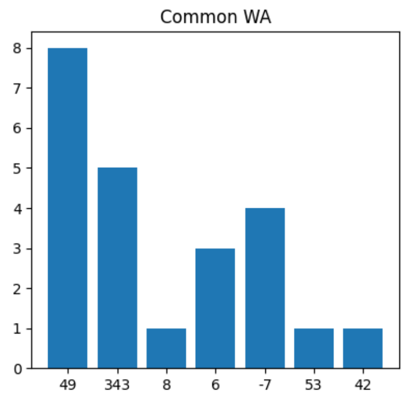
Problem ID: PRATUWS
Group: S1
Problem Type: multiplication
Equation: 2*x + 4 = 18
Answer: 7
A: 2
B: 4
C: 18
N Students Wrong First Problem: 10
N Students Attempted First Problem: 22
First Prob Accuracy: 54.55%
N Students Wrong Overall: 36
N Students Attempted Overall: 147
Problem Overall Accuracy: 75.51%
CWA/(CWA + TOWA): 44.44%



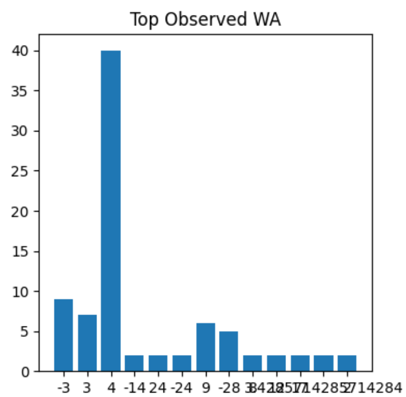


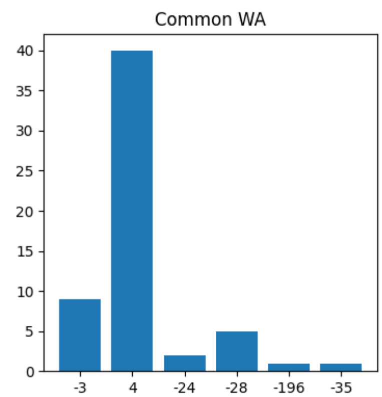
Problem ID: PRATUWT
Group: S1
Problem Type: multiplication
Equation: 11*x + 8 = -25
Answer: -3
A: 11
B: 8
C: -25
N Students Wrong First Problem: 20
N Students Attempted First Problem: 32
First Prob Accuracy: 37.50%
N Students Wrong Overall: 67
N Students Attempted Overall: 161
Problem Overall Accuracy: 58.39%
CWA/(CWA + TOWA): 39.76%



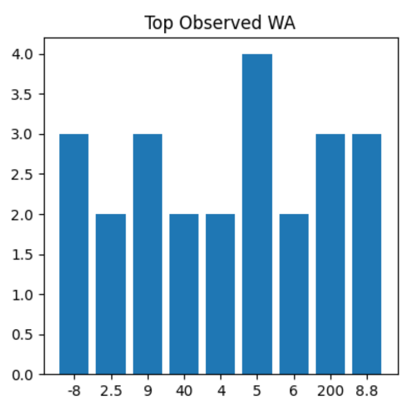


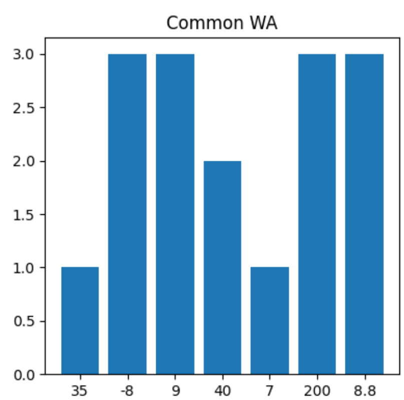
Problem ID: PRATUWU
Group: S1
Problem Type: multiplication
Equation: 7*x + 2 = 51
Answer: 7
A: 7
B: 2
C: 51
N Students Wrong First Problem: 3
N Students Attempted First Problem: 23
First Prob Accuracy: 86.96%
N Students Wrong Overall: 24
N Students Attempted Overall: 158
Problem Overall Accuracy: 84.81%
CWA/(CWA + TOWA): 42.59%



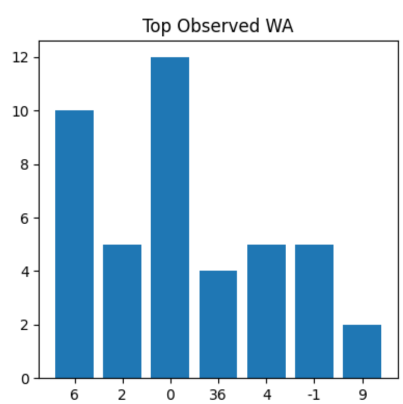


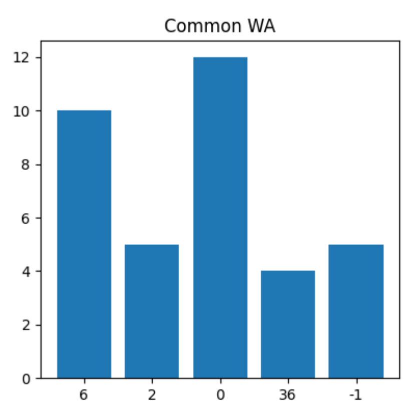
Problem ID: PRATUWV
Group: S1
Problem Type: multiplication
Equation: 7*x + 2 = -26
Answer: -4
A: 7
B: 2
C: -26
N Students Wrong First Problem: 13
N Students Attempted First Problem: 25
First Prob Accuracy: 48.00%
N Students Wrong Overall: 63
N Students Attempted Overall: 155
Problem Overall Accuracy: 59.35%
CWA/(CWA + TOWA): 41.13%



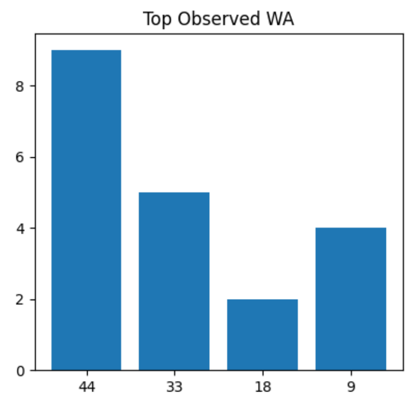


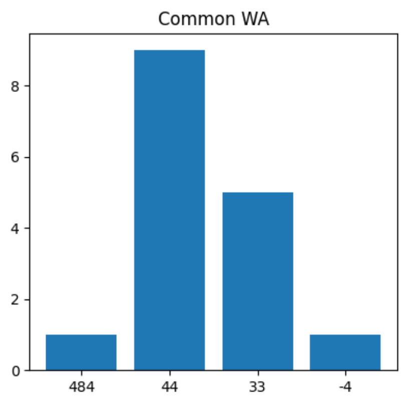
Problem ID: PRATUWW
Group: S1
Problem Type: multiplication
Equation: 5*x + 2 = 42
Answer: 8
A: 5
B: 2
C: 42
N Students Wrong First Problem: 6
N Students Attempted First Problem: 29
First Prob Accuracy: 79.31%
N Students Wrong Overall: 19
N Students Attempted Overall: 149
Problem Overall Accuracy: 87.25%
CWA/(CWA + TOWA): 40.00%



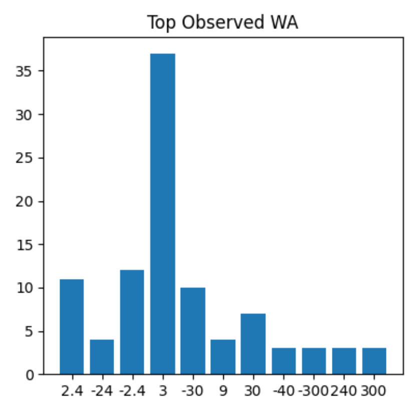


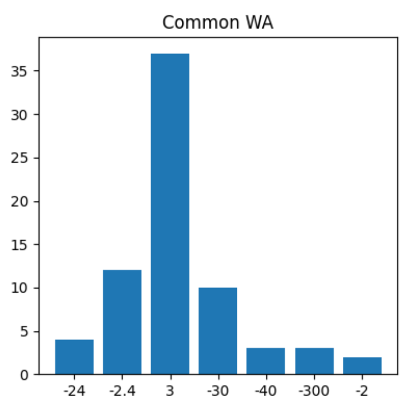
Problem ID: PRATUWX
Group: S1
Problem Type: multiplication
Equation: 6*x + 8 = 14
Answer: 1
A: 6
B: 8
C: 14
N Students Wrong First Problem: 9
N Students Attempted First Problem: 25
First Prob Accuracy: 64.00%
N Students Wrong Overall: 36
N Students Attempted Overall: 169
Problem Overall Accuracy: 78.70%
CWA/(CWA + TOWA): 45.57%



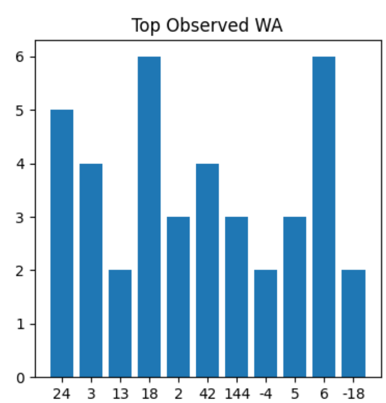


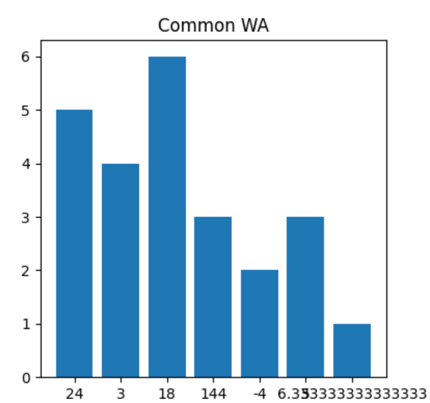
Problem ID: PRATUWY
Group: S1
Problem Type: multiplication
Equation: 11*x + 7 = 51
Answer: 4
A: 11
B: 7
C: 51
N Students Wrong First Problem: 8
N Students Attempted First Problem: 26
First Prob Accuracy: 69.23%
N Students Wrong Overall: 25
N Students Attempted Overall: 171
Problem Overall Accuracy: 85.38%
CWA/(CWA + TOWA): 44.44%



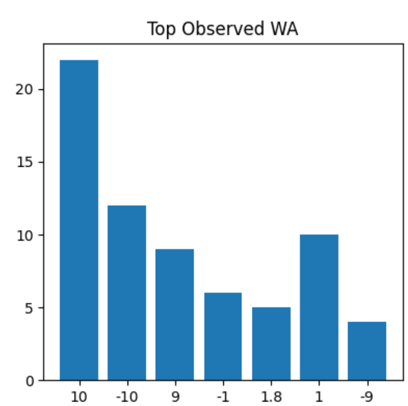


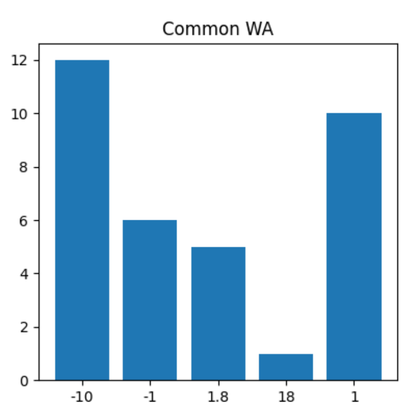
Problem ID: PRATUWZ
Group: S1
Problem Type: multiplication
Equation: 10*x + 3 = -27
Answer: -3
A: 10
B: 3
C: -27
N Students Wrong First Problem: 15
N Students Attempted First Problem: 25
First Prob Accuracy: 40.00%
N Students Wrong Overall: 71
N Students Attempted Overall: 149
Problem Overall Accuracy: 52.35%
CWA/(CWA + TOWA): 42.26%



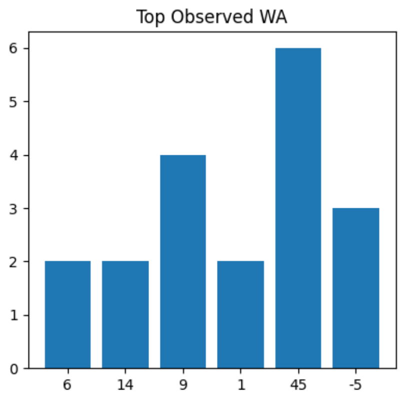


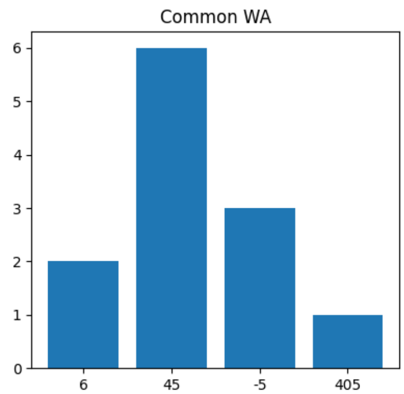
Problem ID: PRATUXA
Group: S1
Problem Type: multiplication
Equation: 6*x + 7 = 31
Answer: 4
A: 6
B: 7
C: 31
N Students Wrong First Problem: 4
N Students Attempted First Problem: 16
First Prob Accuracy: 75.00%
N Students Wrong Overall: 27
N Students Attempted Overall: 158
Problem Overall Accuracy: 82.91%
CWA/(CWA + TOWA): 37.50%



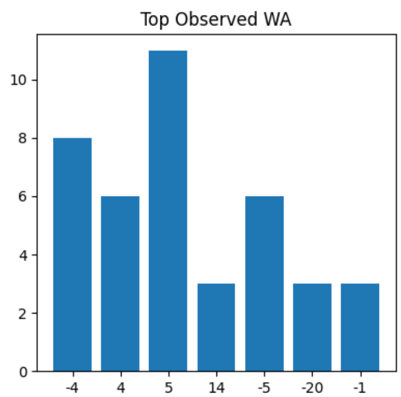


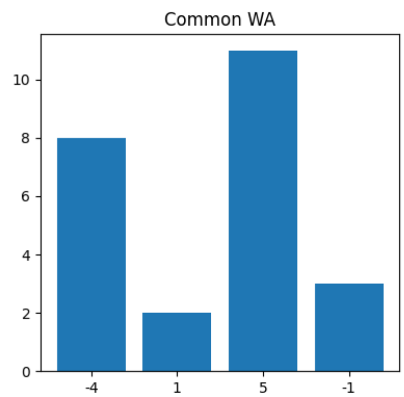
Problem ID: PRATUXB
Group: S1
Problem Type: multiplication
Equation: 10*x + 9 = 9
Answer: 0
A: 10
B: 9
C: 9
N Students Wrong First Problem: 13
N Students Attempted First Problem: 28
First Prob Accuracy: 53.57%
N Students Wrong Overall: 42
N Students Attempted Overall: 163
Problem Overall Accuracy: 74.23%
CWA/(CWA + TOWA): 33.33%



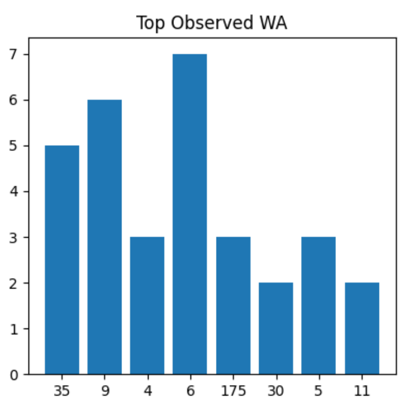


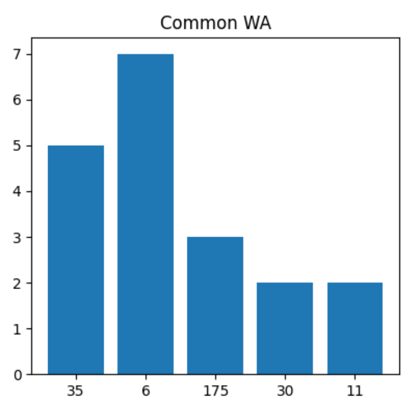
Problem ID: PRATUXC
Group: S1
Problem Type: multiplication
Equation: 9*x + 7 = 52
Answer: 5
A: 9
B: 7
C: 52
N Students Wrong First Problem: 7
N Students Attempted First Problem: 24
First Prob Accuracy: 70.83%
N Students Wrong Overall: 20
N Students Attempted Overall: 158
Problem Overall Accuracy: 87.34%
CWA/(CWA + TOWA): 38.71%



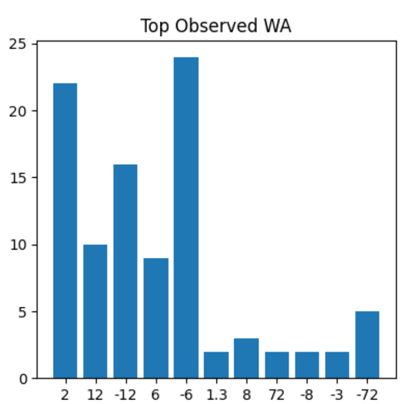


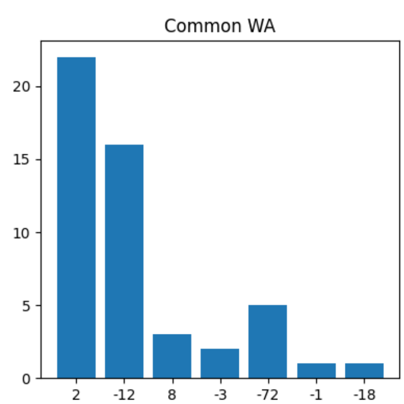
Problem ID: PRATUXD
Group: S1
Problem Type: multiplication
Equation: 4*x + 10 = 10
Answer: 0
A: 4
B: 10
C: 10
N Students Wrong First Problem: 8
N Students Attempted First Problem: 21
First Prob Accuracy: 61.90%
N Students Wrong Overall: 36
N Students Attempted Overall: 167
Problem Overall Accuracy: 78.44%
CWA/(CWA + TOWA): 37.50%



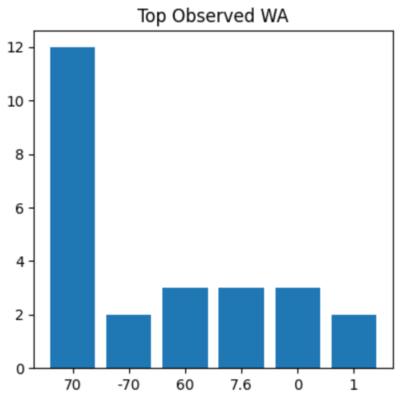


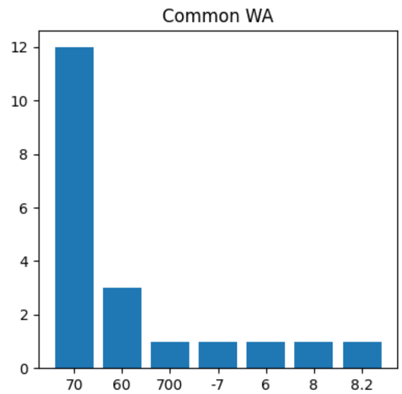
Problem ID: PRATUXE
Group: S1
Problem Type: multiplication
Equation: 5*x + 10 = 45
Answer: 7
A: 5
B: 10
C: 45
N Students Wrong First Problem: 8
N Students Attempted First Problem: 19
First Prob Accuracy: 57.89%
N Students Wrong Overall: 27
N Students Attempted Overall: 161
Problem Overall Accuracy: 83.23%
CWA/(CWA + TOWA): 38.00%



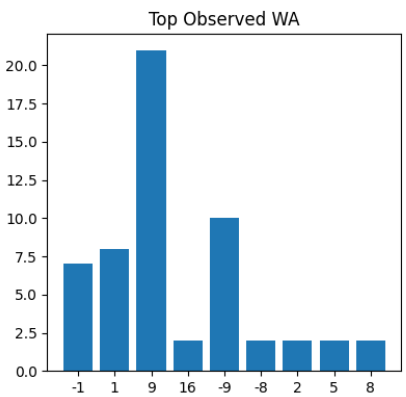


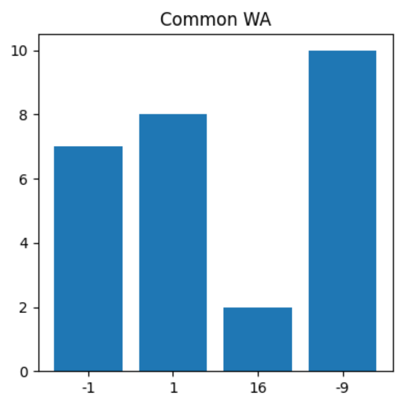
Problem ID: PRATUXF
Group: S1
Problem Type: multiplication
Equation: 6*x + 10 = -2
Answer: -2
A: 6
B: 10
C: -2
N Students Wrong First Problem: 10
N Students Attempted First Problem: 20
First Prob Accuracy: 50.00%
N Students Wrong Overall: 58
N Students Attempted Overall: 143
Problem Overall Accuracy: 59.44%
CWA/(CWA + TOWA): 34.01%



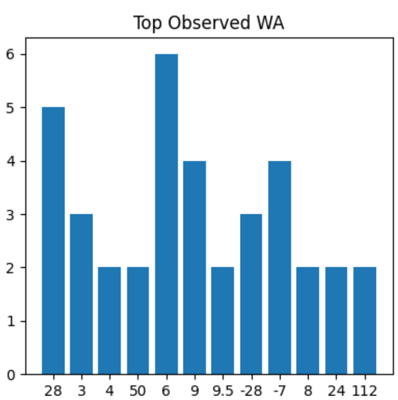


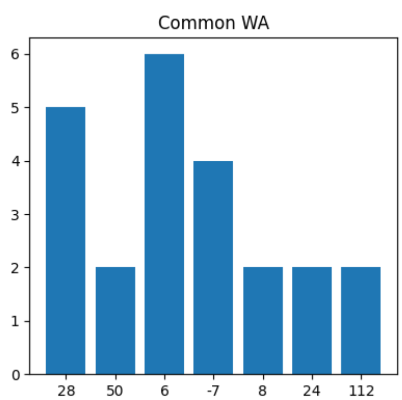
Problem ID: PRATUXG
Group: S1
Problem Type: multiplication
Equation: 10*x + 6 = 76
Answer: 7
A: 10
B: 6
C: 76
N Students Wrong First Problem: 6
N Students Attempted First Problem: 19
First Prob Accuracy: 68.42%
N Students Wrong Overall: 17
N Students Attempted Overall: 137
Problem Overall Accuracy: 87.59%
CWA/(CWA + TOWA): 44.44%



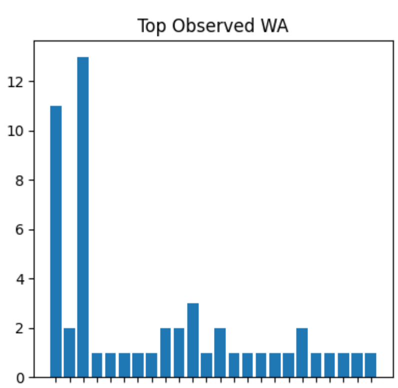


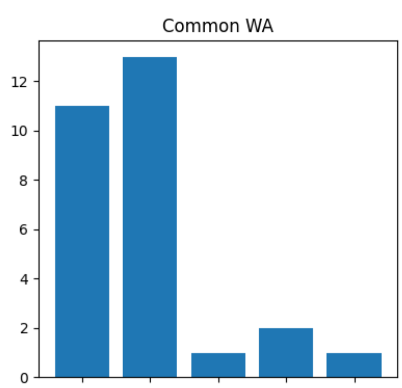
Problem ID: PRATUXH
Group: S1
Problem Type: multiplication
Equation: 9*x + 8 = 8
Answer: 0
A: 9
B: 8
C: 8
N Students Wrong First Problem: 7
N Students Attempted First Problem: 17
First Prob Accuracy: 58.82%
N Students Wrong Overall: 43
N Students Attempted Overall: 150
Problem Overall Accuracy: 71.33%
CWA/(CWA + TOWA): 32.53%



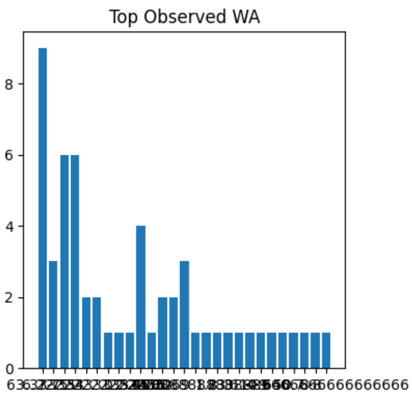


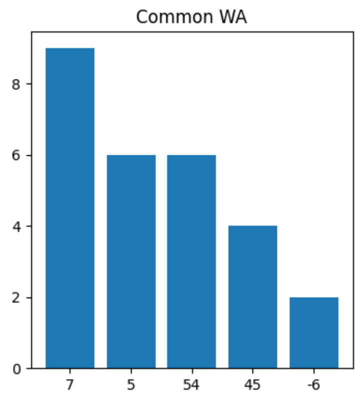
Problem ID: PRATUXJ
Group: S1
Problem Type: multiplication
Equation: 4*x + 11 = 39
Answer: 7
A: 4
B: 11
C: 39
N Students Wrong First Problem: 8
N Students Attempted First Problem: 25
First Prob Accuracy: 68.00%
N Students Wrong Overall: 33
N Students Attempted Overall: 166
Problem Overall Accuracy: 80.12%
CWA/(CWA + TOWA): 38.33%



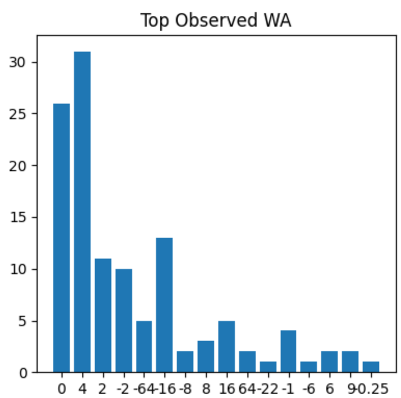


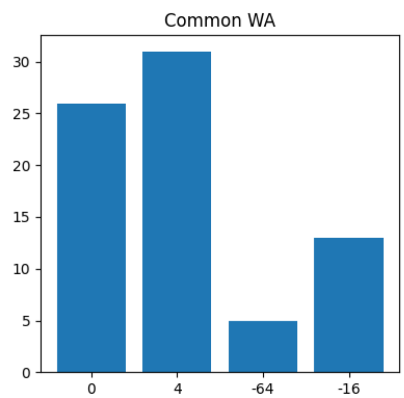
Problem ID: PRA26AA
Group: S2
Problem Type: multiplication
Equation: 10*x + 7 = 27
Answer: 2
A: 10
B: 7
C: 27
N Students Wrong First Problem: 11
N Students Attempted First Problem: 27
First Prob Accuracy: 59.26%
N Students Wrong Overall: 23
N Students Attempted Overall: 149
Problem Overall Accuracy: 84.56%
CWA/(CWA + TOWA): 34.57%



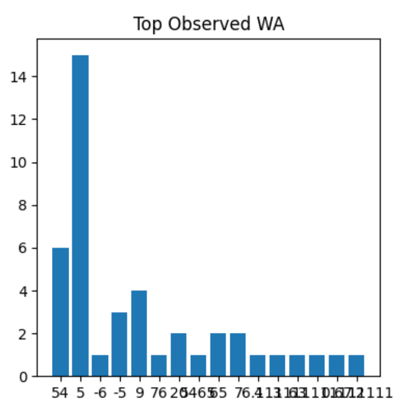


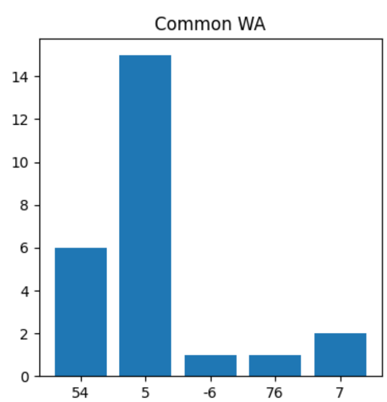
Problem ID: PRA26AB
Group: S2
Problem Type: multiplication
Equation: 9*x + 7 = 61
Answer: 6
A: 9
B: 7
C: 61
N Students Wrong First Problem: 9
N Students Attempted First Problem: 24
First Prob Accuracy: 62.50%
N Students Wrong Overall: 31
N Students Attempted Overall: 164
Problem Overall Accuracy: 81.10%
CWA/(CWA + TOWA): 32.53%



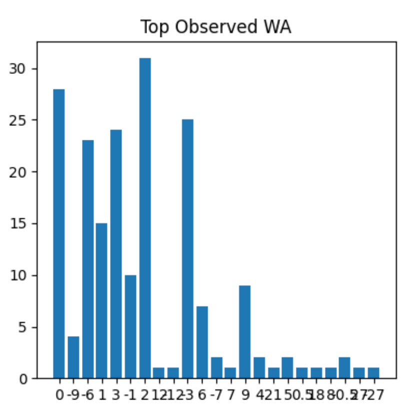


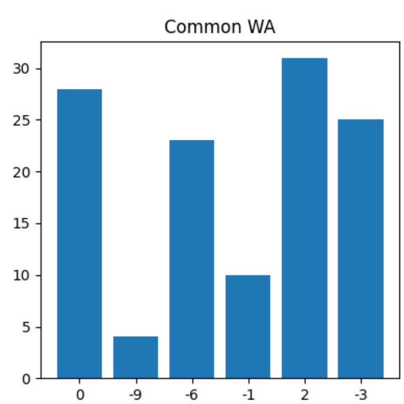
Problem ID: PRA26AC
Group: S2
Problem Type: multiplication
Equation: 4*x + 8 = -8
Answer: -4
A: 4
B: 8
C: -8
N Students Wrong First Problem: 9
N Students Attempted First Problem: 19
First Prob Accuracy: 52.63%
N Students Wrong Overall: 70
N Students Attempted Overall: 171
Problem Overall Accuracy: 59.06%
CWA/(CWA + TOWA): 38.66%



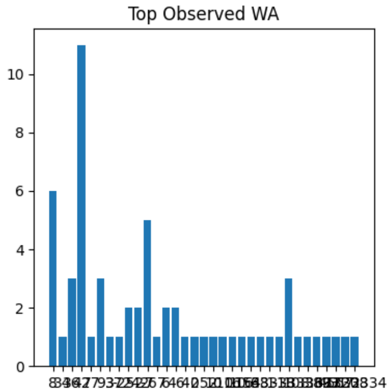


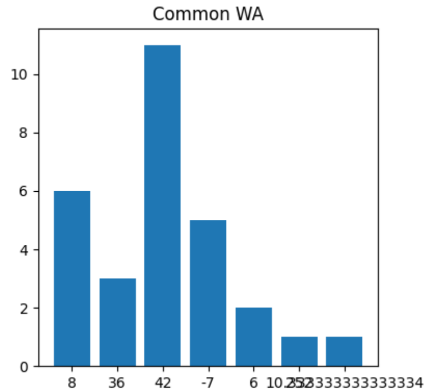
Problem ID: PRA26AD
Group: S2
Problem Type: multiplication
Equation: 9*x + 11 = 65
Answer: 6
A: 9
B: 11
C: 65
N Students Wrong First Problem: 3
N Students Attempted First Problem: 16
First Prob Accuracy: 81.25%
N Students Wrong Overall: 22
N Students Attempted Overall: 164
Problem Overall Accuracy: 86.59%
CWA/(CWA + TOWA): 36.76%



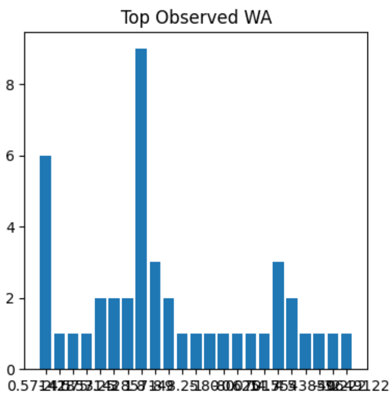


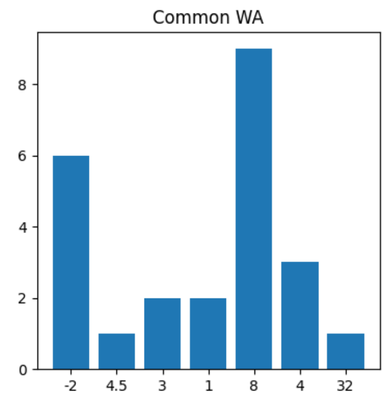
Problem ID: PRA26AE
Group: S2
Problem Type: multiplication
Equation: 3*x + 3 = -3
Answer: -2
A: 3
B: 3
C: -3
N Students Wrong First Problem: 26
N Students Attempted First Problem: 33
First Prob Accuracy: 21.21%
N Students Wrong Overall: 88
N Students Attempted Overall: 184
Problem Overall Accuracy: 52.17%
CWA/(CWA + TOWA): 38.54%



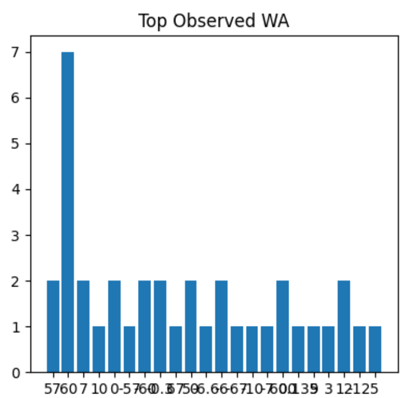


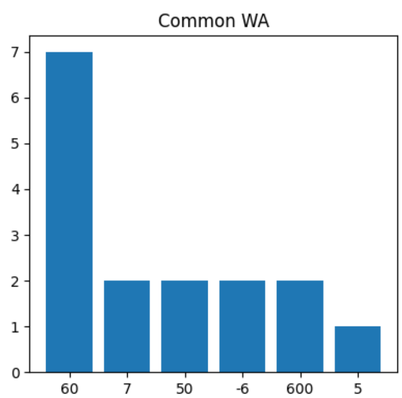
Problem ID: PRA26AF
Group: S2
Problem Type: multiplication
Equation: 6*x + 10 = 52
Answer: 7
A: 6
B: 10
C: 52
N Students Wrong First Problem: 13
N Students Attempted First Problem: 26
First Prob Accuracy: 50.00%
N Students Wrong Overall: 29
N Students Attempted Overall: 164
Problem Overall Accuracy: 82.32%
CWA/(CWA + TOWA): 31.87%



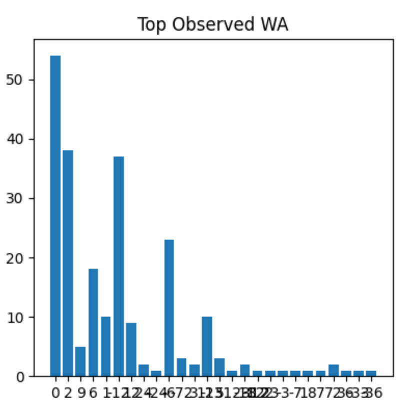


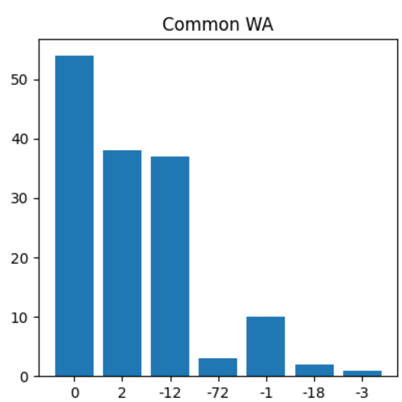
Problem ID: PRA26AG
Group: S2
Problem Type: multiplication
Equation: 4*x + 5 = 13
Answer: 2
A: 4
B: 5
C: 13
N Students Wrong First Problem: 7
N Students Attempted First Problem: 26
First Prob Accuracy: 73.08%
N Students Wrong Overall: 30
N Students Attempted Overall: 147
Problem Overall Accuracy: 79.59%
CWA/(CWA + TOWA): 34.78%



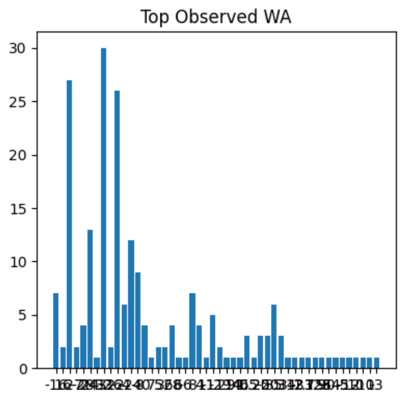


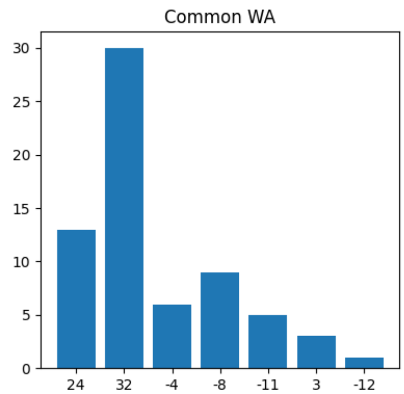
Problem ID: PRA26AH
Group: S2
Problem Type: multiplication
Equation: 10*x + 7 = 67
Answer: 6
A: 10
B: 7
C: 67
N Students Wrong First Problem: 3
N Students Attempted First Problem: 23
First Prob Accuracy: 86.96%
N Students Wrong Overall: 15
N Students Attempted Overall: 140
Problem Overall Accuracy: 89.29%
CWA/(CWA + TOWA): 30.19%



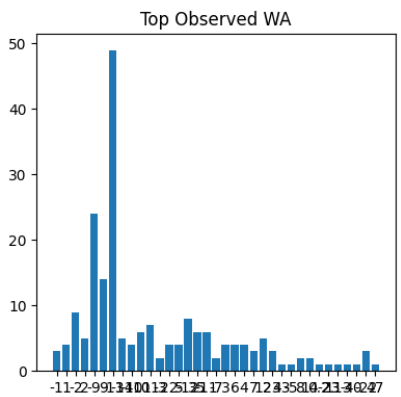


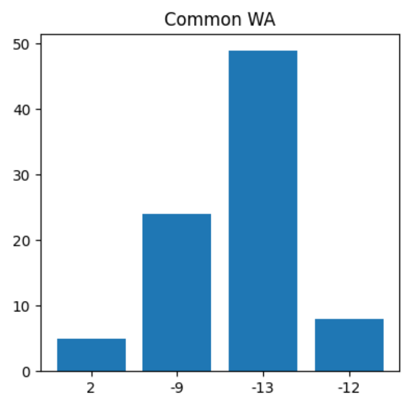
Problem ID: PRA26AJ
Group: S2
Problem Type: multiplication
Equation: 6*x + 6 = -6
Answer: -2
A: 6
B: 6
C: -6
N Students Wrong First Problem: 28
N Students Attempted First Problem: 34
First Prob Accuracy: 17.65%
N Students Wrong Overall: 84
N Students Attempted Overall: 175
Problem Overall Accuracy: 52.00%
CWA/(CWA + TOWA): 38.77%



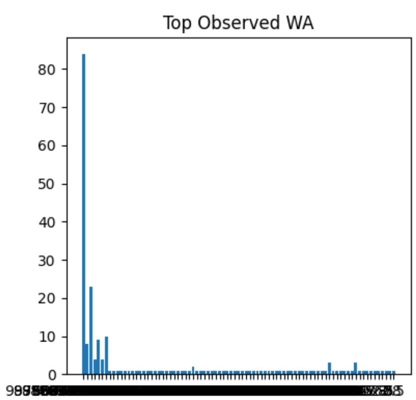


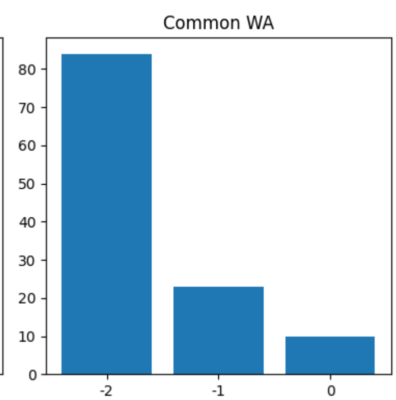
Problem ID: PRA26B2
Group: S2
Problem Type: division
Equation: x/4 + 7 = -1
Answer: -32
A: 4
B: 7
C: -1
N Students Wrong First Problem: 20
N Students Attempted First Problem: 30
First Prob Accuracy: 33.33%
N Students Wrong Overall: 87
N Students Attempted Overall: 170
Problem Overall Accuracy: 48.82%
CWA/(CWA + TOWA): 24.10%



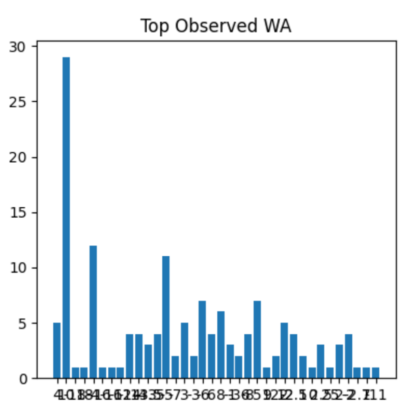


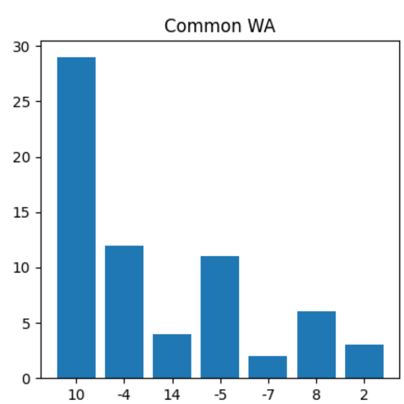
Problem ID: PRA26B3
Group: S2
Problem Type: division
Equation: x/-1 + 11 = -2
Answer: 13
A: -1
B: 11
C: -2
N Students Wrong First Problem: 24
N Students Attempted First Problem: 30
First Prob Accuracy: 20.00%
N Students Wrong Overall: 90
N Students Attempted Overall: 170
Problem Overall Accuracy: 47.06%
CWA/(CWA + TOWA): 30.07%



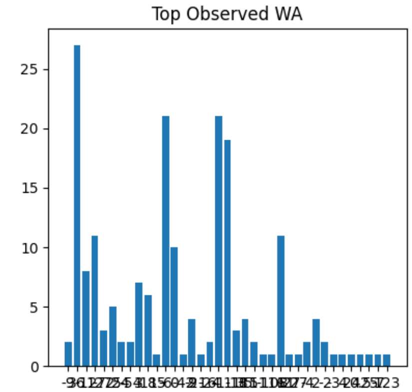


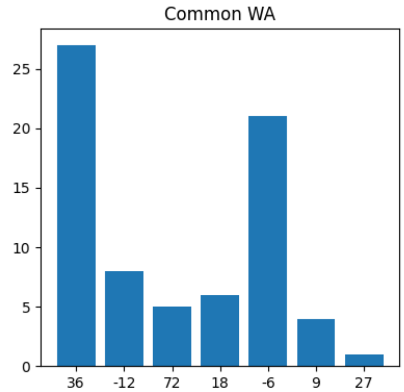
Problem ID: PRA26B4
Group: S2
Problem Type: division
Equation: x/-1 + 2 = 0
Answer: 2
A: -1
B: 2
C: 0
N Students Wrong First Problem: 24
N Students Attempted First Problem: 32
First Prob Accuracy: 25.00%
N Students Wrong Overall: 85
N Students Attempted Overall: 167
Problem Overall Accuracy: 49.10%
CWA/(CWA + TOWA): 34.41%



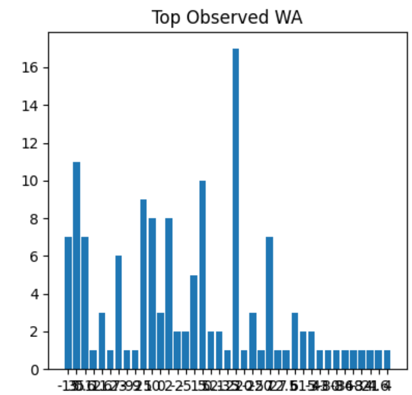


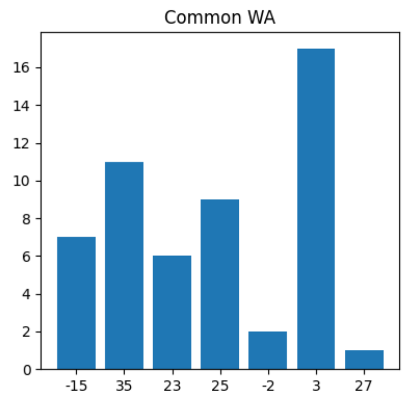
Problem ID: PRA26B6
Group: S2
Problem Type: division
Equation: x/2 + 6 = 1
Answer: -10
A: 2
B: 6
C: 1
N Students Wrong First Problem: 19
N Students Attempted First Problem: 30
First Prob Accuracy: 36.67%
N Students Wrong Overall: 69
N Students Attempted Overall: 161
Problem Overall Accuracy: 57.14%
CWA/(CWA + TOWA): 31.16%



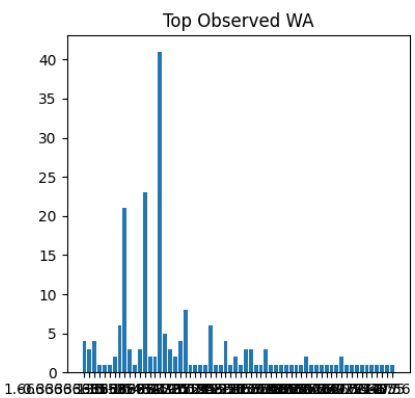


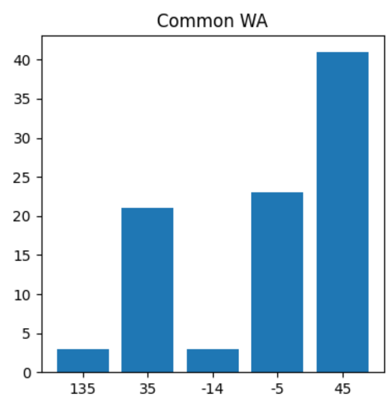
Problem ID: PRA26BC
Group: S2
Problem Type: division
Equation: x/6 + 9 = 3
Answer: -36
A: 6
B: 9
C: 3
N Students Wrong First Problem: 19
N Students Attempted First Problem: 21
First Prob Accuracy: 9.52%
N Students Wrong Overall: 78
N Students Attempted Overall: 165
Problem Overall Accuracy: 52.73%
CWA/(CWA + TOWA): 27.27%



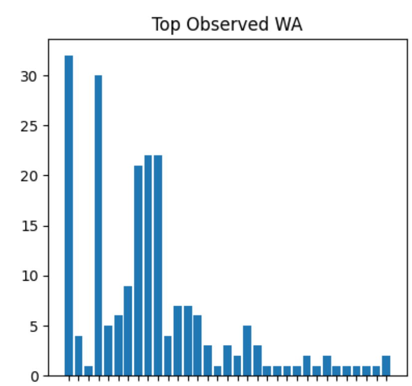


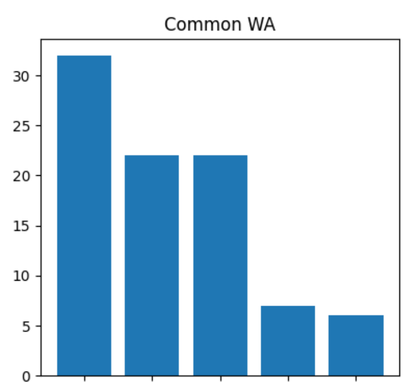
Problem ID: PRA26BD
Group: S2
Problem Type: division
Equation: x/5 + 2 = 5
Answer: 15
A: 5
B: 2
C: 5
N Students Wrong First Problem: 12
N Students Attempted First Problem: 21
First Prob Accuracy: 42.86%
N Students Wrong Overall: 59
N Students Attempted Overall: 160
Problem Overall Accuracy: 63.12%
CWA/(CWA + TOWA): 27.89%



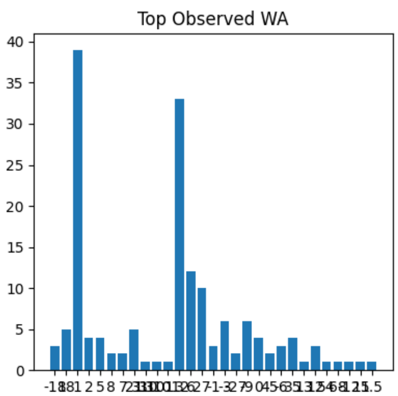


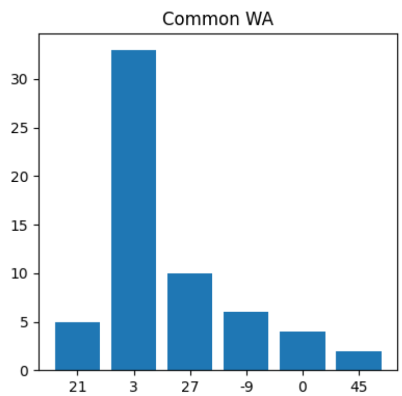
Problem ID: PRA26BE
Group: S2
Problem Type: division
Equation: x/9 + 10 = 5
Answer: -45
A: 9
B: 10
C: 5
N Students Wrong First Problem: 24
N Students Attempted First Problem: 28
First Prob Accuracy: 14.29%
N Students Wrong Overall: 92
N Students Attempted Overall: 164
Problem Overall Accuracy: 43.90%
CWA/(CWA + TOWA): 31.49%



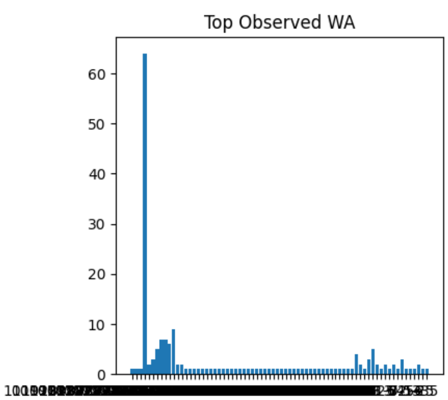


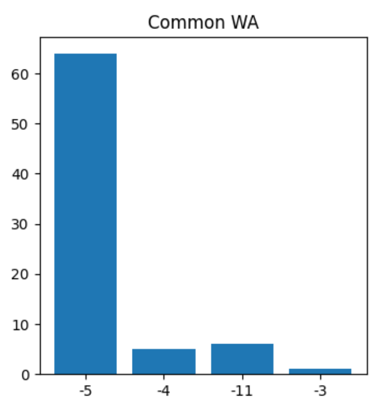
Problem ID: PRA26BF
Group: S2
Problem Type: division
Equation: x/2 + 6 = 4
Answer: -4
A: 2
B: 6
C: 4
N Students Wrong First Problem: 24
N Students Attempted First Problem: 31
First Prob Accuracy: 22.58%
N Students Wrong Overall: 87
N Students Attempted Overall: 159
Problem Overall Accuracy: 45.28%
CWA/(CWA + TOWA): 29.87%



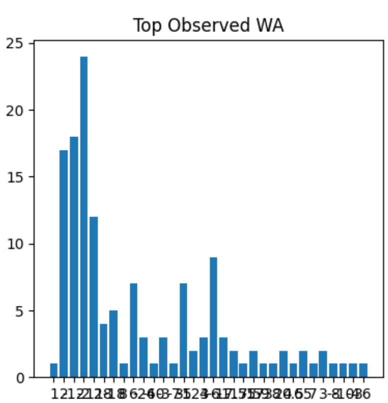


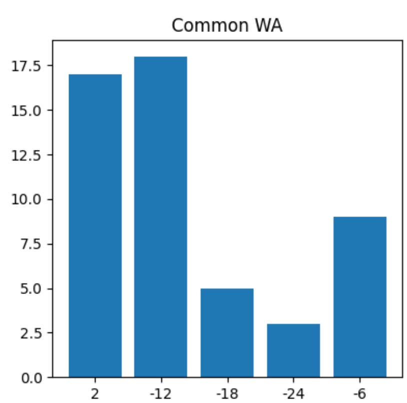
Problem ID: PRA26BG
Group: S2
Problem Type: division
Equation: x/3 + 6 = 9
Answer: 9
A: 3
B: 6
C: 9
N Students Wrong First Problem: 22
N Students Attempted First Problem: 30
First Prob Accuracy: 26.67%
N Students Wrong Overall: 83
N Students Attempted Overall: 200
Problem Overall Accuracy: 58.50%
CWA/(CWA + TOWA): 27.15%



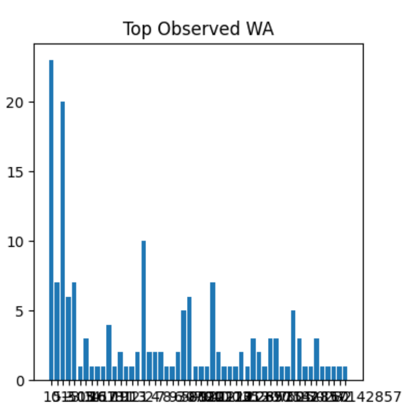


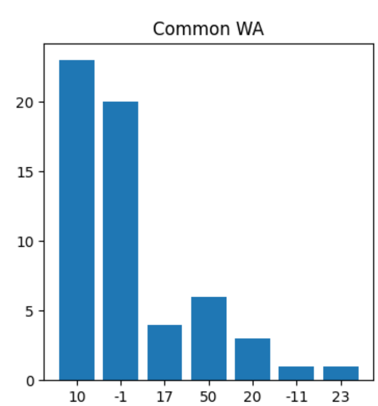
Problem ID: PRA26BH
Group: S2
Problem Type: division
Equation: x/-1 + 8 = 3
Answer: 5
A: -1
B: 8
C: 3
N Students Wrong First Problem: 24
N Students Attempted First Problem: 35
First Prob Accuracy: 31.43%
N Students Wrong Overall: 86
N Students Attempted Overall: 198
Problem Overall Accuracy: 56.57%
CWA/(CWA + TOWA): 29.12%



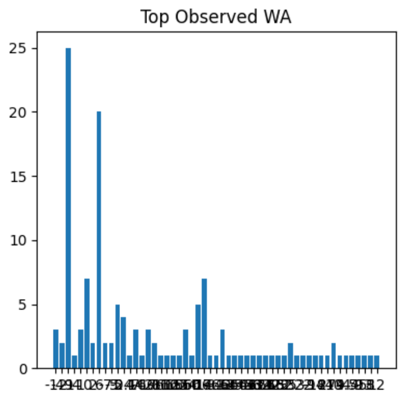


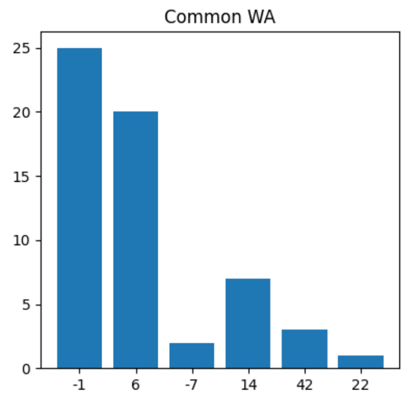
Problem ID: PRA26BJ
Group: S2
Problem Type: division
Equation: x/-2 + 6 = 6
Answer: 0
A: -2
B: 6
C: 6
N Students Wrong First Problem: 10
N Students Attempted First Problem: 21
First Prob Accuracy: 52.38%
N Students Wrong Overall: 62
N Students Attempted Overall: 166
Problem Overall Accuracy: 62.65%
CWA/(CWA + TOWA): 27.08%



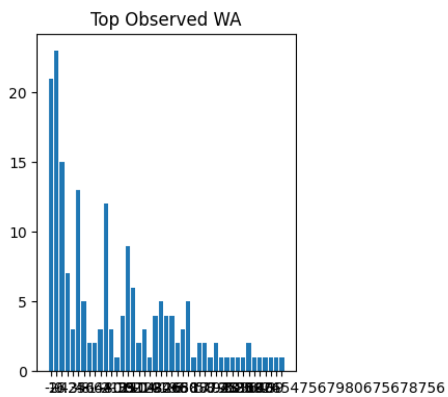


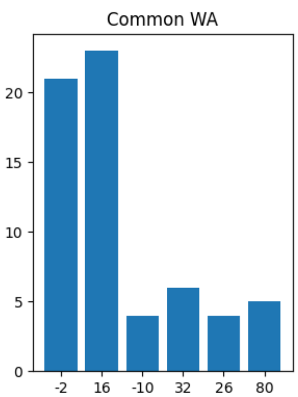
Problem ID: PRA26BK
Group: S2
Problem Type: division
Equation: x/10 + 3 = 2
Answer: -10
A: 10
B: 3
C: 2
N Students Wrong First Problem: 21
N Students Attempted First Problem: 30
First Prob Accuracy: 30.00%
N Students Wrong Overall: 64
N Students Attempted Overall: 154
Problem Overall Accuracy: 58.44%
CWA/(CWA + TOWA): 26.36%



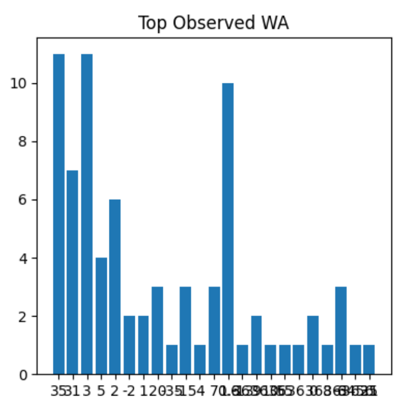


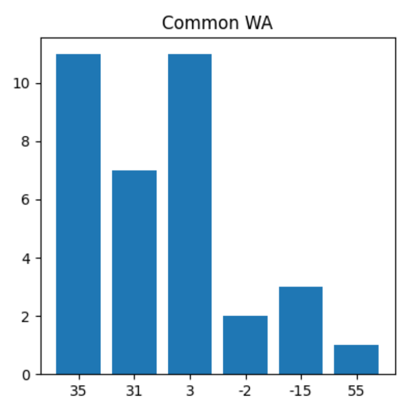
Problem ID: PRA26BM
Group: S2
Problem Type: division
Equation: x/6 + 4 = 3
Answer: -6
A: 6
B: 4
C: 3
N Students Wrong First Problem: 22
N Students Attempted First Problem: 30
First Prob Accuracy: 26.67%
N Students Wrong Overall: 63
N Students Attempted Overall: 151
Problem Overall Accuracy: 58.28%
CWA/(CWA + TOWA): 29.59%



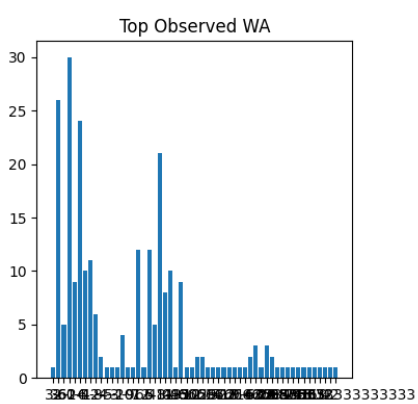


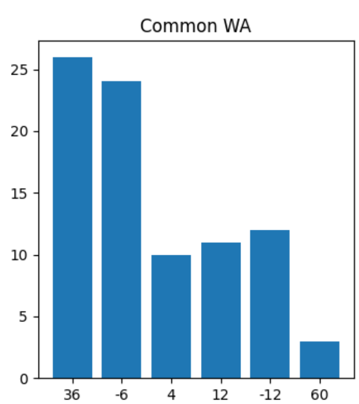
Problem ID: PRA26BN
Group: S2
Problem Type: division
Equation: x/8 + 6 = 4
Answer: -16
A: 8
B: 6
C: 4
N Students Wrong First Problem: 23
N Students Attempted First Problem: 3
First Prob Accuracy: 28.12%
N Students Wrong Overall: 82
N Students Attempted Overall: 159
Problem Overall Accuracy: 48.43%
CWA/(CWA + TOWA): 25.61%



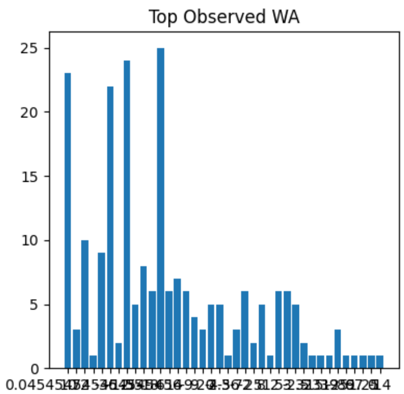


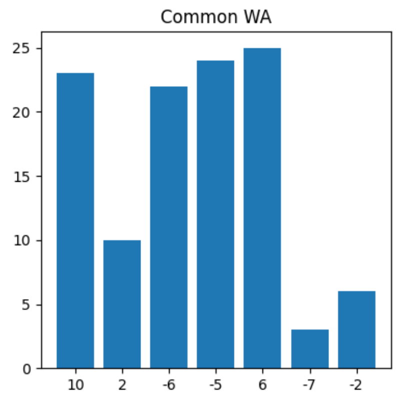
Problem ID: PRA26BP
Group: S2
Problem Type: division
Equation: x/5 + 4 = 7
Answer: 15
A: 5
B: 4
C: 7
N Students Wrong First Problem: 13
N Students Attempted First Problem: 20
First Prob Accuracy: 35.00%
N Students Wrong Overall: 44
N Students Attempted Overall: 159
Problem Overall Accuracy: 72.33%
CWA/(CWA + TOWA): 30.97%



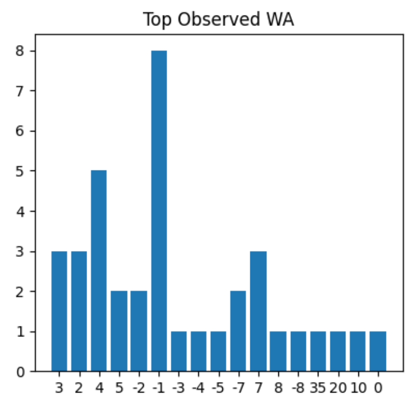


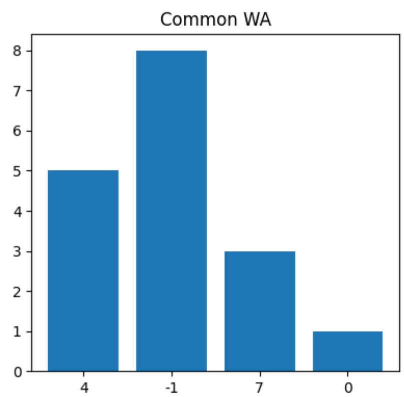
Problem ID: PRA26BQ
Group: S2
Problem Type: division
Equation: x/6 + 8 = 2
Answer: -36
A: 6
B: 8
C: 2
N Students Wrong First Problem: 17
N Students Attempted First Problem: 24
First Prob Accuracy: 29.17%
N Students Wrong Overall: 95
N Students Attempted Overall: 173
Problem Overall Accuracy: 45.09%
CWA/(CWA + TOWA): 25.67%



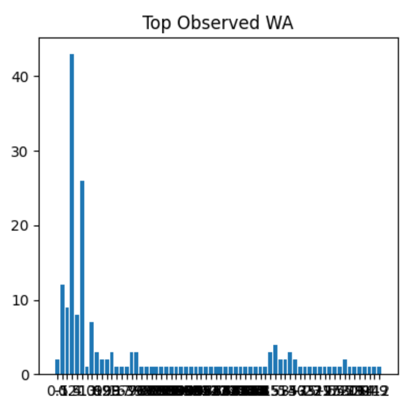


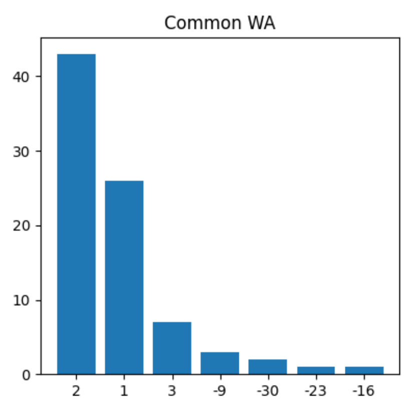
Problem ID: PRA26BR
Group: S2
Problem Type: division
Equation: x/2 + 4 = -1
Answer: -10
A: 2
B: 4
C: -1
N Students Wrong First Problem: 22
N Students Attempted First Problem: 29
First Prob Accuracy: 24.14%
N Students Wrong Overall: 88
N Students Attempted Overall: 176
Problem Overall Accuracy: 50.00%
CWA/(CWA + TOWA): 33.73%



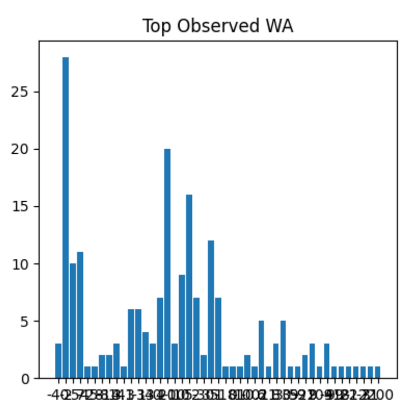


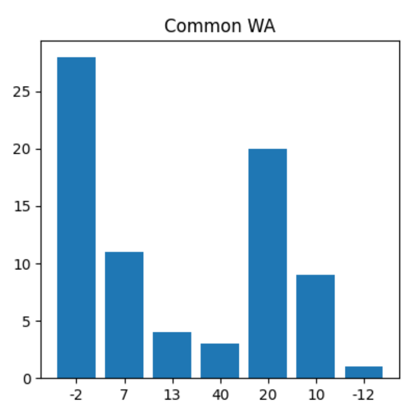
Problem ID: PRA26BS
Group: S2
Problem Type: division
Equation: x/1 + 3 = 4
Answer: 1
A: 1
B: 3
C: 4
N Students Wrong First Problem: 4
N Students Attempted First Problem: 25
First Prob Accuracy: 84.00%
N Students Wrong Overall: 20
N Students Attempted Overall: 138
Problem Overall Accuracy: 85.51%
CWA/(CWA + TOWA): 31.48%



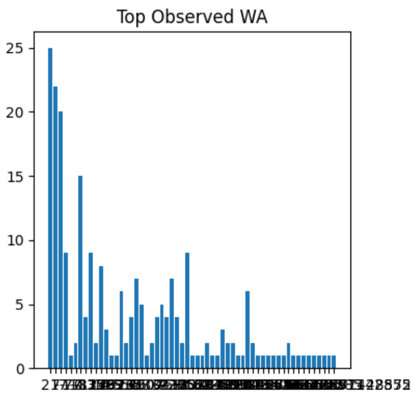


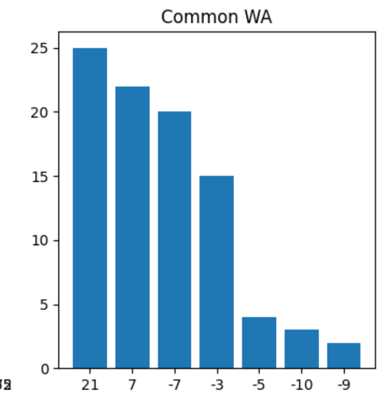
Problem ID: PRA26BT
Group: S2
Problem Type: division
Equation: x/-2 + 7 = 8
Answer: -2
A: -2
B: 7
C: 8
N Students Wrong First Problem: 16
N Students Attempted First Problem: 21
First Prob Accuracy: 23.81%
N Students Wrong Overall: 80
N Students Attempted Overall: 188
Problem Overall Accuracy: 57.45%
CWA/(CWA + TOWA): 30.74%



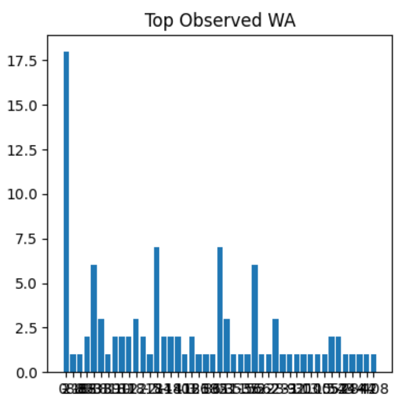


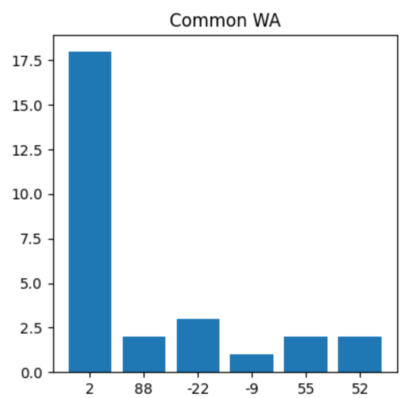
Problem ID: PRA26BU
Group: S2
Problem Type: division
Equation: x/10 + 3 = 1
Answer: -20
A: 10
B: 3
C: 1
N Students Wrong First Problem: 21
N Students Attempted First Problem: 27
First Prob Accuracy: 22.22%
N Students Wrong Overall: 72
N Students Attempted Overall: 165
Problem Overall Accuracy: 56.36%
CWA/(CWA + TOWA): 27.34%



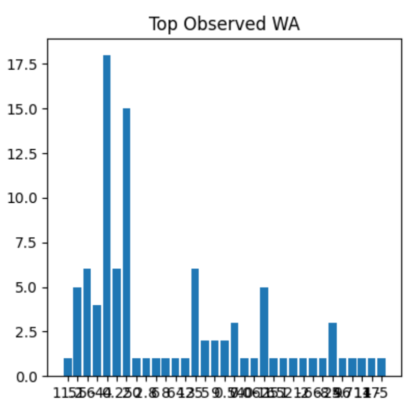


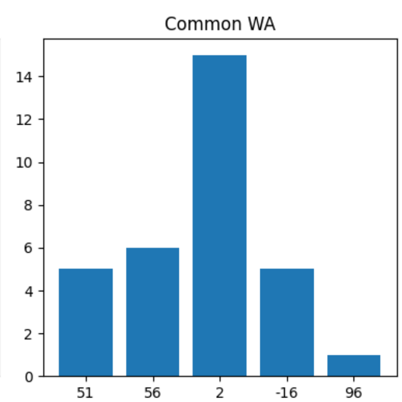
Problem ID: PRA26BV
Group: S2
Problem Type: division
Equation: x/7 + 2 = -1
Answer: -21
A: 7
B: 2
C: -1
N Students Wrong First Problem: 27
N Students Attempted First Problem: 33
First Prob Accuracy: 18.18%
N Students Wrong Overall: 84
N Students Attempted Overall: 174
Problem Overall Accuracy: 51.72%
CWA/(CWA + TOWA): 28.80%



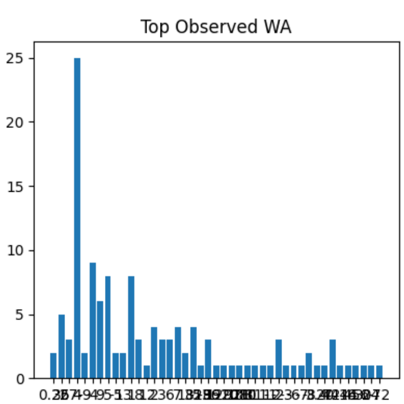


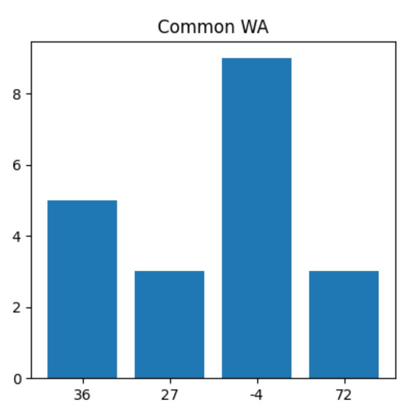
Problem ID: PRA26BW
Group: S2
Problem Type: division
Equation: x/11 + 3 = 5
Answer: 22
A: 11
B: 3
C: 5
N Students Wrong First Problem: 12
N Students Attempted First Problem: 23
First Prob Accuracy: 47.83%
N Students Wrong Overall: 37
N Students Attempted Overall: 154
Problem Overall Accuracy: 75.97%
CWA/(CWA + TOWA): 21.37%



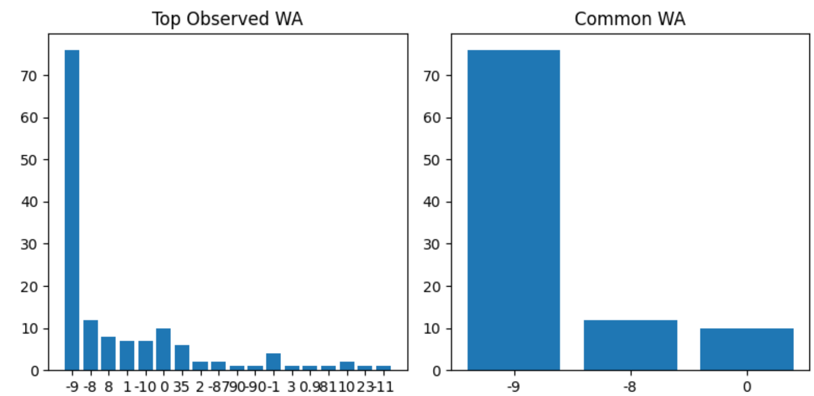


Problem ID: PRA26BX
Group: S2
Problem Type: division
Equation: x/8 + 5 = 7
Answer: 16
A: 8
B: 5
C: 7
N Students Wrong First Problem: 18
N Students Attempted First Problem: 28
First Prob Accuracy: 35.71%
N Students Wrong Overall: 53
N Students Attempted Overall: 163
Problem Overall Accuracy: 67.48%
CWA/(CWA + TOWA): 24.81%

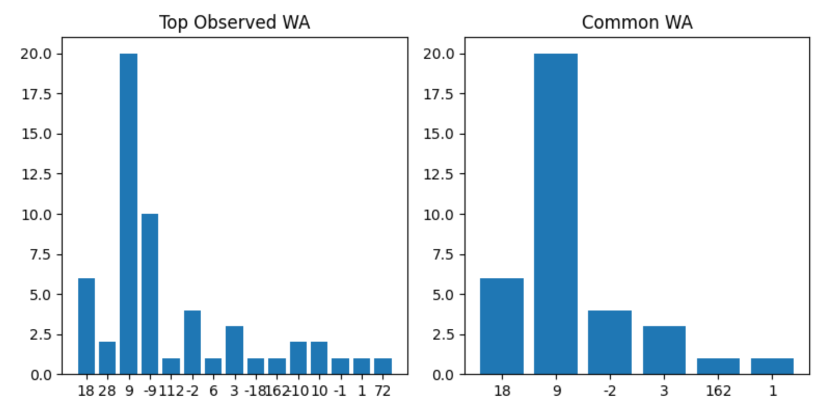




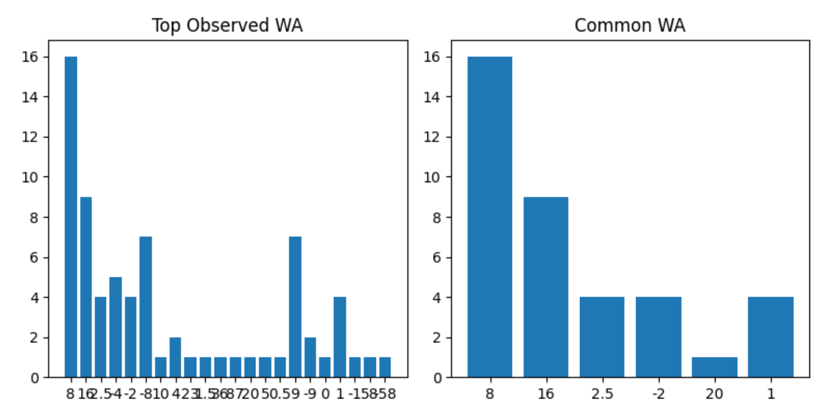
Problem ID: PRA26BY
Group: S2
Problem Type: division
Equation: x/4 + 9 = 9
Answer: 0
A: 4
B: 9
C: 9
N Students Wrong First Problem: 15
N Students Attempted First Problem: 31
First Prob Accuracy: 51.61%
N Students Wrong Overall: 48
N Students Attempted Overall: 154
Problem Overall Accuracy: 68.83%
CWA/(CWA + TOWA): 13.61%



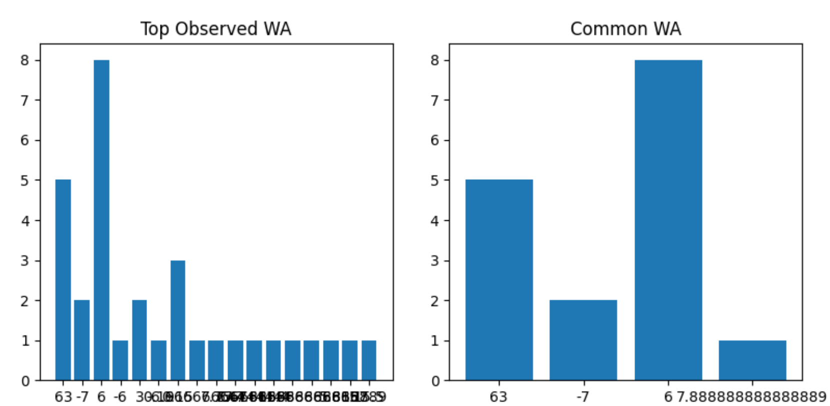
Problem ID: PRA26BZ
Group: S2
Problem Type: division
Equation: x/-1 + 9 = 0
Answer: 9
A: -1
B: 9
C: 0
N Students Wrong First Problem: 19
N Students Attempted First Problem: 29
First Prob Accuracy: 34.48%
N Students Wrong Overall: 96
N Students Attempted Overall: 183
Problem Overall Accuracy: 47.54%
CWA/(CWA + TOWA): 40.66%



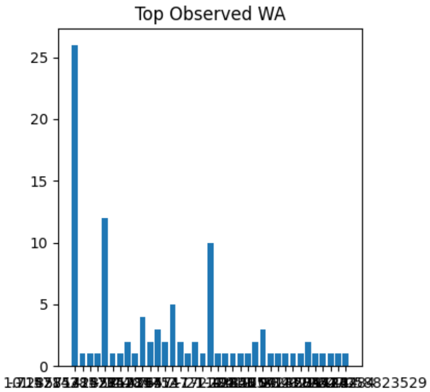
Problem ID: PRA259R
Group: S2
Problem Type: multiplication
Equation: 9*x + 10 = 28
Answer: 2
A: 9
B: 10
C: 28
N Students Wrong First Problem: 6
N Students Attempted First Problem: 22
First Prob Accuracy: 72.73%
N Students Wrong Overall: 27
N Students Attempted Overall: 153
Problem Overall Accuracy: 82.35%
CWA/(CWA + TOWA): 38.46%

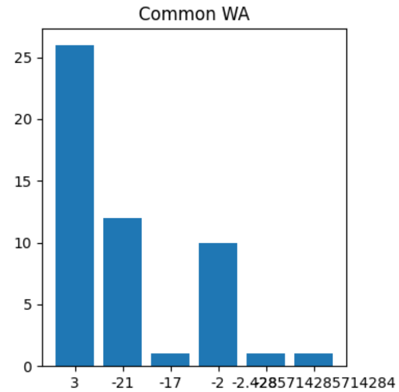


Problem ID: PRA259S
Group: S2
Problem Type: multiplication
Equation: 8*x + 2 = 18
Answer: 2
A: 8
B: 2
C: 18
N Students Wrong First Problem: 13
N Students Attempted First Problem: 34
First Prob Accuracy: 61.76%
N Students Wrong Overall: 30
N Students Attempted Overall: 165
Problem Overall Accuracy: 81.82%
CWA/(CWA + TOWA): 34.55%

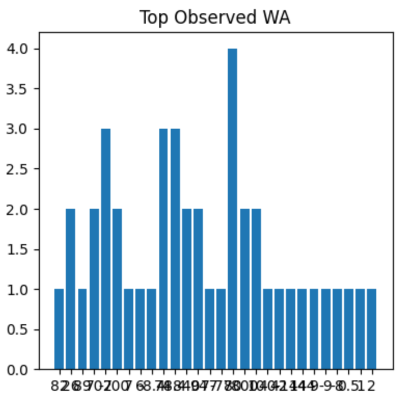


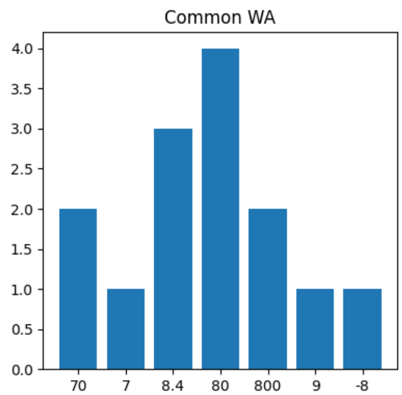
Problem ID: PRA259T
Group: S2
Problem Type: multiplication
Equation: 9*x + 4 = 67
Answer: 7
A: 9
B: 4
C: 67
N Students Wrong First Problem: 4
N Students Attempted First Problem: 25
First Prob Accuracy: 84.00%
N Students Wrong Overall: 24
N Students Attempted Overall: 163
Problem Overall Accuracy: 85.28%
CWA/(CWA + TOWA): 33.33%



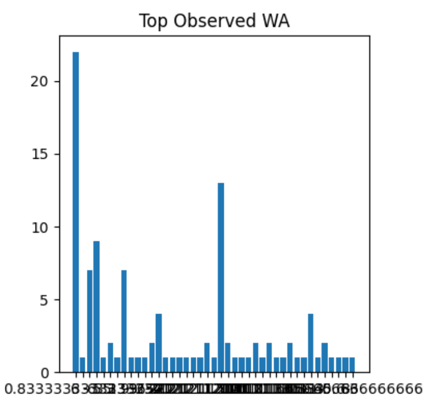


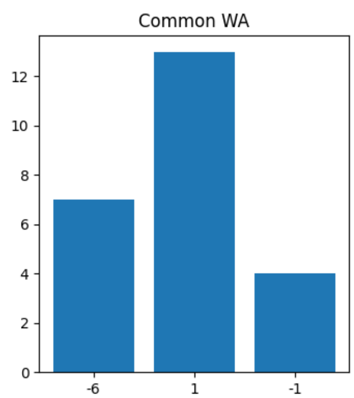
Problem ID: PRA259U
Group: S2
Problem Type: multiplication
Equation: 7*x + 2 = -19
Answer: -3
A: 7
B: 2
C: -19
N Students Wrong First Problem: 12
N Students Attempted First Problem: 23
First Prob Accuracy: 47.83%
N Students Wrong Overall: 48
N Students Attempted Overall: 145
Problem Overall Accuracy: 66.90%
CWA/(CWA + TOWA): 33.77%



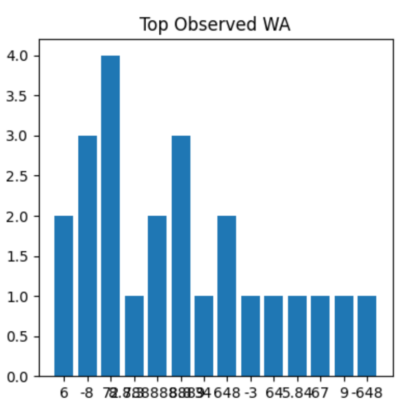


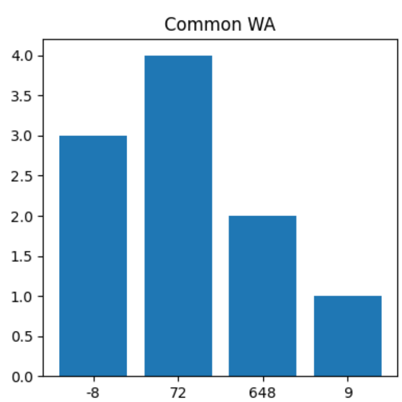
Problem ID: PRA259V
Group: S2
Problem Type: multiplication
Equation: 10*x + 2 = 82
Answer: 8
A: 10
B: 2
C: 82
N Students Wrong First Problem: 6
N Students Attempted First Problem: 23
First Prob Accuracy: 73.91%
N Students Wrong Overall: 17
N Students Attempted Overall: 173
Problem Overall Accuracy: 90.17%
CWA/(CWA + TOWA): 24.14%



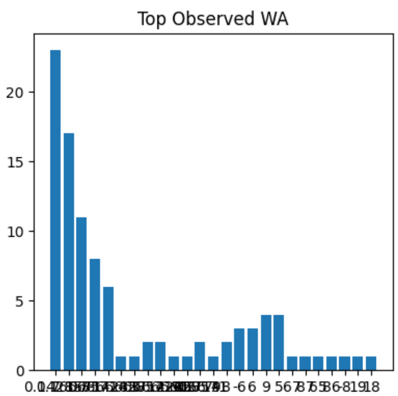


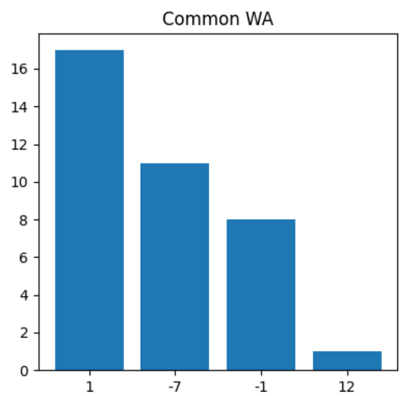
Problem ID: PRA259W
Group: S2
Problem Type: multiplication
Equation: 6*x + 5 = 5
Answer: 0
A: 6
B: 5
C: 5
N Students Wrong First Problem: 12
N Students Attempted First Problem: 26
First Prob Accuracy: 53.85%
N Students Wrong Overall: 45
N Students Attempted Overall: 177
Problem Overall Accuracy: 74.58%
CWA/(CWA + TOWA): 18.18%



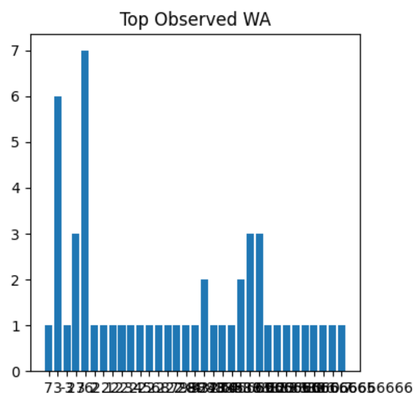


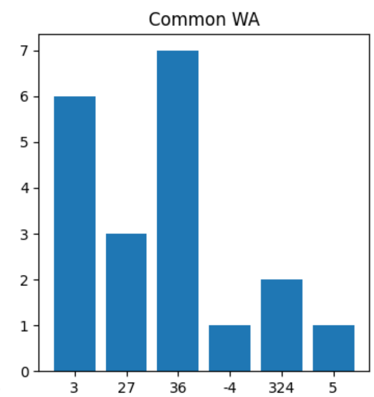
Problem ID: PRA259X
Group: S2
Problem Type: multiplication
Equation: 9*x + 4 = 76
Answer: 8
A: 9
B: 4
C: 76
N Students Wrong First Problem: 6
N Students Attempted First Problem: 23
First Prob Accuracy: 73.91%
N Students Wrong Overall: 17
N Students Attempted Overall: 144
Problem Overall Accuracy: 88.19%
CWA/(CWA + TOWA): 29.41%



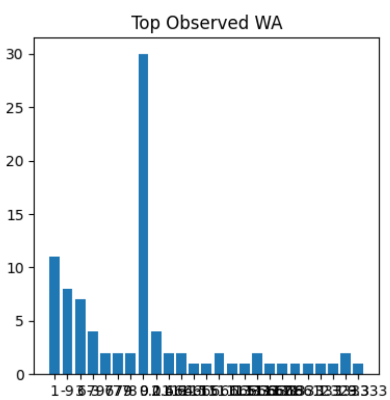


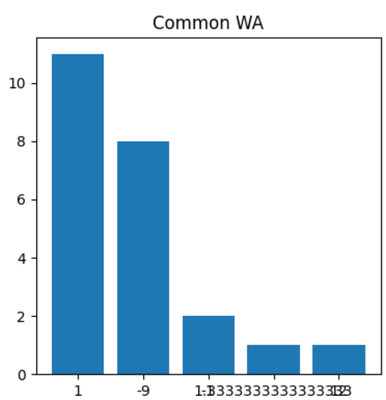
Problem ID: PRA259Y
Group: S2
Problem Type: multiplication
Equation: 7*x + 6 = 6
Answer: 0
A: 7
B: 6
C: 6
N Students Wrong First Problem: 6
N Students Attempted First Problem: 21
First Prob Accuracy: 71.43%
N Students Wrong Overall: 47
N Students Attempted Overall: 176
Problem Overall Accuracy: 73.30%
CWA/(CWA + TOWA): 27.21%



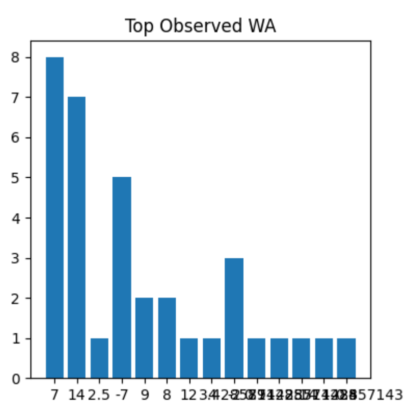


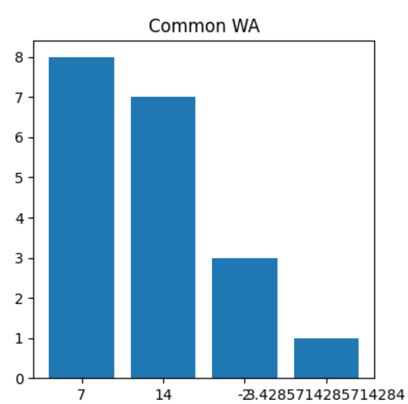
Problem ID: PRA259Z
Group: S2
Problem Type: multiplication
Equation: 9*x + 3 = 39
Answer: 4
A: 9
B: 3
C: 39
N Students Wrong First Problem: 9
N Students Attempted First Problem: 19
First Prob Accuracy: 52.63%
N Students Wrong Overall: 29
N Students Attempted Overall: 159
Problem Overall Accuracy: 81.76%
CWA/(CWA + TOWA): 27.78%



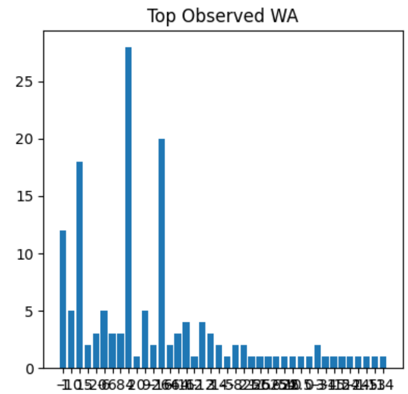


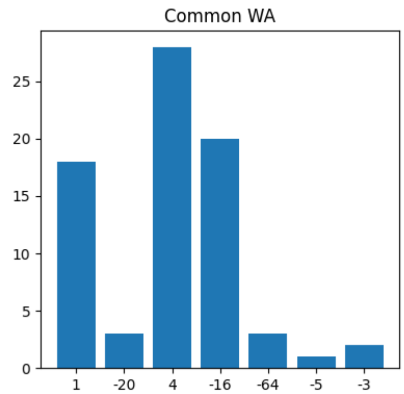
Problem ID: PRA2592
Group: S2
Problem Type: multiplication
Equation: 9*x + 6 = 6
Answer: 0
A: 9
B: 6
C: 6
N Students Wrong First Problem: 14
N Students Attempted First Problem: 29
First Prob Accuracy: 51.72%
N Students Wrong Overall: 48
N Students Attempted Overall: 177
Problem Overall Accuracy: 72.88%
CWA/(CWA + TOWA): 20.18%



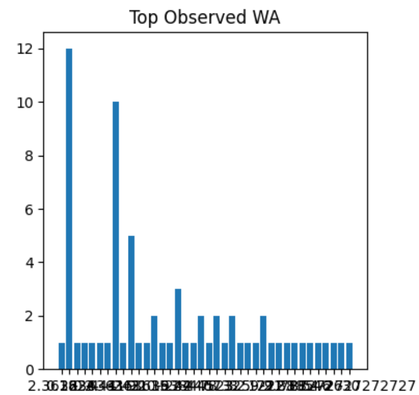


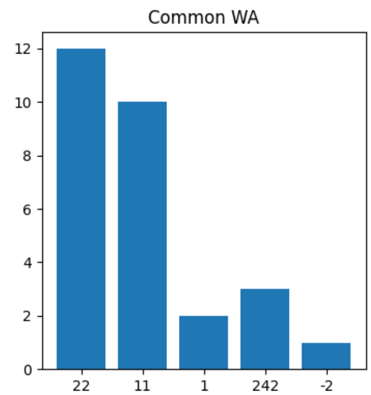
Problem ID: PRA2593
Group: S2
Problem Type: multiplication
Equation: 7*x + 5 = 19
Answer: 2
A: 7
B: 5
C: 19
N Students Wrong First Problem: 2
N Students Attempted First Problem: 11
First Prob Accuracy: 81.82%
N Students Wrong Overall: 21
N Students Attempted Overall: 152
Problem Overall Accuracy: 86.18%
CWA/(CWA + TOWA): 35.19%



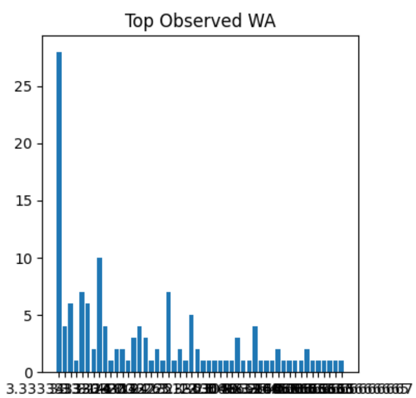


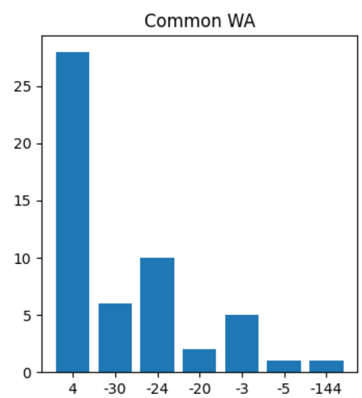
Problem ID: PRA2594
Group: S2
Problem Type: multiplication
Equation: 4*x + 10 = -6
Answer: -4
A: 4
B: 10
C: -6
N Students Wrong First Problem: 15
N Students Attempted First Problem: 27
First Prob Accuracy: 44.44%
N Students Wrong Overall: 70
N Students Attempted Overall: 169
Problem Overall Accuracy: 58.58%
CWA/(CWA + TOWA): 33.48%



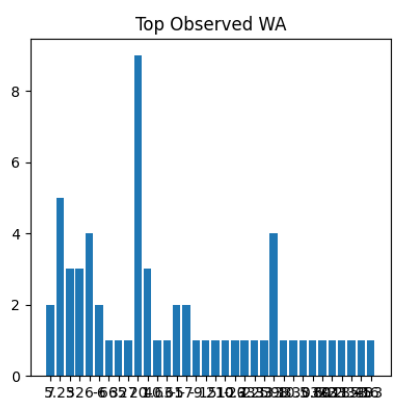


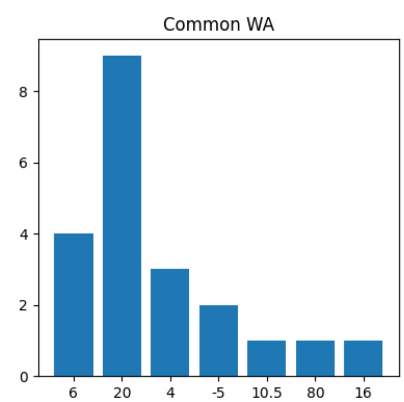
Problem ID: PRA2595
Group: S2
Problem Type: multiplication
Equation: 11*x + 9 = 31
Answer: 2
A: 11
B: 9
C: 31
N Students Wrong First Problem: 8
N Students Attempted First Problem: 22
First Prob Accuracy: 63.64%
N Students Wrong Overall: 33
N Students Attempted Overall: 164
Problem Overall Accuracy: 79.88%
CWA/(CWA + TOWA): 28.87%



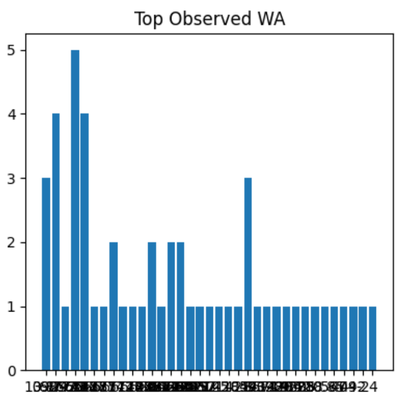


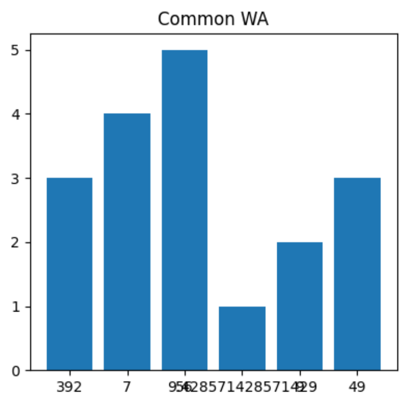
Problem ID: PRA2596
Group: S2
Problem Type: multiplication
Equation: 6*x + 2 = -22
Answer: -4
A: 6
B: 2
C: -22
N Students Wrong First Problem: 17
N Students Attempted First Problem: 28
First Prob Accuracy: 39.29%
N Students Wrong Overall: 53
N Students Attempted Overall: 168
Problem Overall Accuracy: 68.45%
CWA/(CWA + TOWA): 27.75%



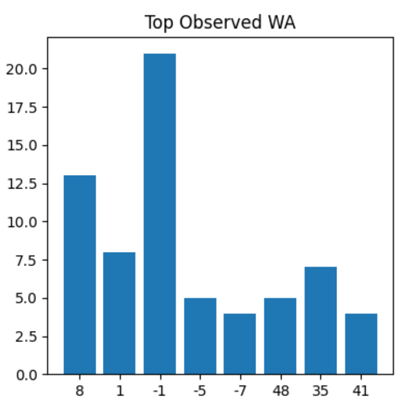


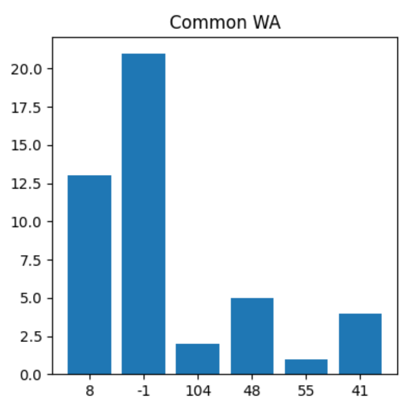
Problem ID: PRA2597
Group: S2
Problem Type: multiplication
Equation: 4*x + 11 = 31
Answer: 5
A: 4
B: 11
C: 31
N Students Wrong First Problem: 8
N Students Attempted First Problem: 24
First Prob Accuracy: 66.67%
N Students Wrong Overall: 25
N Students Attempted Overall: 159
Problem Overall Accuracy: 84.28%
CWA/(CWA + TOWA): 25.30%



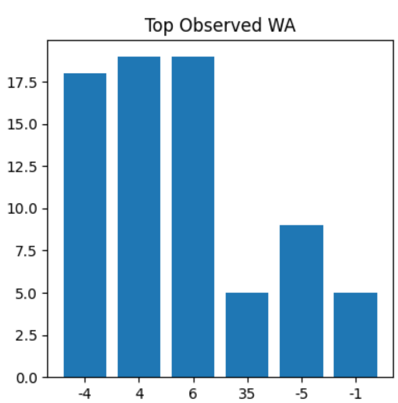


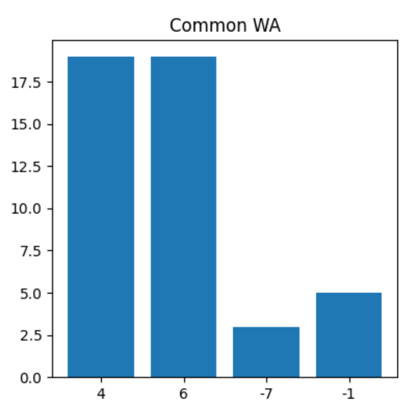
Problem ID: PRA2599
Group: S2
Problem Type: multiplication
Equation: 7*x + 5 = 61
Answer: 8
A: 7
B: 5
C: 61
N Students Wrong First Problem: 5
N Students Attempted First Problem: 41
First Prob Accuracy: 87.80%
N Students Wrong Overall: 29
N Students Attempted Overall: 189
Problem Overall Accuracy: 84.66%
CWA/(CWA + TOWA): 25.35%



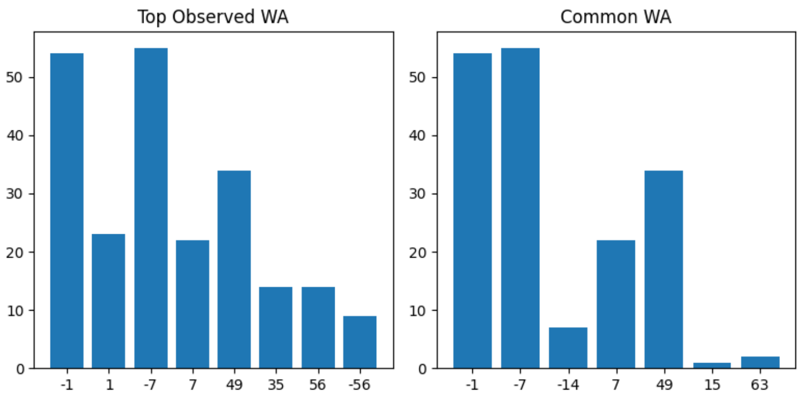


Problem ID: PRATU5A
Group: S1
Problem Type: division
Equation: x/8 + 7 = 6
Answer: -8
A: 8
B: 7
C: 6
N Students Wrong First Problem: 15
N Students Attempted First Problem: 27
First Prob Accuracy: 44.44%
N Students Wrong Overall: 54
N Students Attempted Overall: 161
Problem Overall Accuracy: 66.46%
CWA/(CWA + TOWA): 40.71%

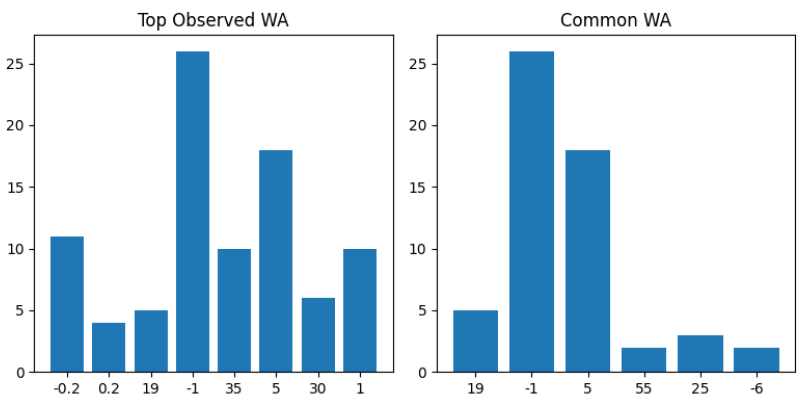




Problem ID: PRATU5B
Group: S1
Problem Type: division
Equation: x/1 + 5 = -1
Answer: -6
A: 1
B: 5
C: -1
N Students Wrong First Problem: 9
N Students Attempted First Problem: 25
First Prob Accuracy: 64.00%
N Students Wrong Overall: 54
N Students Attempted Overall: 171
Problem Overall Accuracy: 68.42%
CWA/(CWA + TOWA): 38.02%



Problem ID: PRATU5D
Group: S1
Problem Type: division
Equation: x/7 + 8 = 1
Answer: -49
A: 7
B: 8
C: 1
N Students Wrong First Problem: 42
N Students Attempted First Problem: 49
First Prob Accuracy: 14.29%
N Students Wrong Overall: 133
N Students Attempted Overall: 211
Problem Overall Accuracy: 36.97%
CWA/(CWA + TOWA): 43.75%



Problem ID: PRATU5E
Group: S1
Problem Type: division
Equation: x/5 + 6 = 5
Answer: -5
A: 5
B: 6
C: 5
N Students Wrong First Problem: 20
N Students Attempted First Problem: 29
First Prob Accuracy: 31.03%
N Students Wrong Overall: 66
N Students Attempted Overall: 157
Problem Overall Accuracy: 57.96%
CWA/(CWA + TOWA): 38.36%

Appendix H: 1.0 Messages

Example division problem with corresponding mistake messages

$$\frac{a}{3} + 9 = 5$$

✗ 12

Check your sign

Positive x Positive = Positive
Negative x Negative = Positive
Positive x Negative = Negative
Negative x Positive = Negative

Step 1: Correct
a/3 + 9 = 5
- 9 - 9

Step 2: Correct
3 * a/3 = -4 * 3

Sign: Incorrect
-4 * 3 IS NOT 12

✗ 42

SUBTRACT 9 from both sides
The sign of 9 is addition (+)
Use the opposite sign, subtraction (-), to eliminate 9 from the right side of the equation

Step 1: Incorrect
a/3 + 9 = 5
+ 9 + 9

✗ 6

Subtract 9 from 5 before multiplying by 3
You need to isolate a / 3

Step 1: Incorrect
3 * a/3 + 9 = 5 * 3

✗ 15

Don't forget to subtract 9 before multiplying

Step 1: Incorrect
a/3 + 9 = 5

✗ -4

Don't forget to multiply by 3

Step 1: Correct
a/3 + 9 = 5
- 9 - 9

Step 2: Incorrect
a/3 = -4

✘ -7

MULTIPLY both sides by 3
The operation between a and 3 is division
Use the opposite operation (multiplication in this case) to isolate a

Step 1: Correct
a/3 + 9 = 5
- 9 - 9

Step 2: Incorrect
a/3 = -4
-3 -3

✘ 24

SUBTRACT 9 from 5 before multiplying by 3
You need to isolate a / 3

Step 1: Incorrect
3 * a/3 + 9 = 5 *** 3**

Example multiplication problem with corresponding mistake messages

11a + 9 = 31

✗ -2

Check your sign
Positive / Positive = Positive
Negative / Negative = Positive
Positive / Negative = Negative
Negative / Positive = Negative

Step 1: Correct
11a + 9 = 31
-9 -9

Step 2: Correct
11a = 22
11 11

Sign: Incorrect
a = 22/11 IS NOT -2

✗ 22

Don't forget to divide by 11

Step 1: Correct
11a + 9 = 31
-9 -9

Step 2: Incorrect
11a = 22

✗ 242

DIVIDE both sides by 11
The operation between 11 and a is multiplication
Use the opposite operation (division in this case) to isolate a

Step 1: Correct
11a + 9 = 31
-9 -9

Step 2: Incorrect
11 * 11a = 22 * 11

✗ 3.63636363636364

SUBTRACT 9 from 31
The sign in front of 9 is addition (+)
Use the opposite sign, subtraction (-), to eliminate 9 from the right side of the equation

Step 1: Incorrect
11a + 9 = 31
+9 +9

✗ 11

DIVIDE both sides by 11
The operation between 11 and a is multiplication
Use the opposite operation (division in this case) to isolate a

Step 1: Correct
11a + 9 = 31
-9 -9

Step 2: Incorrect
11a = 22
-11 -11

✘ 40

SUBTRACT 9 from 31

The sign of 9 is addition (+)

Use the opposite sign, subtraction (-), to eliminate 9 from the right side of the equation

Don't forget to divide by 11

Step 1: Incorrect

11a + 9 = 31

+9 +9

✘ 2 + 1.0

Check your division

Step 1: Correct

11a + 9 = 31

-9 -9

Step 2: Correct

11a = 22

11 11

Division: Incorrect

a = 22/11 IS NOT 3

✘ 2 - 1.0

Check your division

Step 1: Correct

11a + 9 = 31

-9 -9

Step 2: Correct

11a = 22

11 11

Division: Incorrect

a = 22/11 IS NOT 1

Appendix I: 2.0 Messages

Example division problem

Solve for a

$$\frac{a}{6} + 4 = 3$$

Corresponding mistake messages without sentiment

✘ 6

We're guessing you did these first steps correctly:

Step 1: Correct

a/6 + 4 = 3

- 4 - 4

Step 2: Correct

6 * a/6 = -1 * 6

But we're guessing this last step, you may have made an error:

Sign: Incorrect

-1 * 6 IS NOT 6

✘ 42

We're guessing you may have made an error on this first step:

Step 1: Incorrect: Remember to subtract 4 from both sides

a/6 + 4 = 3

+ 4 + 4

✘ 14

We're guessing you may have made an error on this first step

Step 1: Incorrect: Make sure to distribute the 6 to the 4 as well

6 * a/6 + 4 = 3 * **6**

✘ 18

We're guessing you may have made an error on this first step

Step 1: Incorrect: Don't forget to subtract 4 from both sides

a/6 + **4** = 3

✘ -1

We're guessing you did this first step correctly:

Step 1: Correct

a/6 + 4 = 3

- 4 - 4

But we're guessing this last step, you may have made an error:

Step 2: Incorrect: Don't forget to multiply both sides by 6

a/**6** = -1

✘ -7

We're guessing you did this first step correctly:

Step 1: Correct

a/6 + 4 = 3

- 4 - 4

But we're guessing this last step, you may have made an error:

Step 2: Incorrect

a/6 = -1

-6 **-6**

✘ 22

We're guessing you may have made an error on this first step:

Step 1: Incorrect: Remember to multiply 4 by 6 as well.

6 * a/6 + 4 = 3 * 6

✘ -1/6

We're guessing you did this first step correctly:

Step 1: Correct

c/6 + 4 = 3

- 4 - 4

But we're guessing this last step, you may have made an error:

Step 2: Incorrect: Make sure to multiply instead of dividing both sides by 6.

c/6 = -1

÷6 ÷6

Corresponding mistake messages with sentiment

✘ 6

You're almost there,

We're guessing you did these first steps correctly:

Step 1: Correct

a/6 + 4 = 3

- 4 - 4

Step 2: Correct

6 * a/6 = -1 * 6

But we're guessing this last step, you may have made an error:

Sign: Incorrect

-1 * 6 IS NOT 6

✘ 42

Whoops,

We're guessing you may have made an error on this first step:

Step 1: Incorrect: Remember to subtract 4 from both sides

a/6 + 4 = 3

+ 4 + 4

✘ 14

You're on the right track,

We're guessing you may have made an error on this first step

Step 1: Incorrect: Make sure to distribute the 6 to the 4 as well

6 * a/6 + 4 = 3 * 6

✘ 18

Whoops,

We're guessing you may have made an error on this first step

Step 1: Incorrect: Don't forget to subtract 4 from both sides

a/6 + 4 = 3

✘ -1

A lot of students make this mistake,

We're guessing you did this first step correctly:

Step 1: Correct

a/6 + 4 = 3

- 4 - 4

But we're guessing this last step, you may have made an error:

Step 2: Incorrect: Don't forget to multiply both sides by 6

a/6 = -1

✘ -7

Your on the right track,

We're guessing you did this first step correctly:

Step 1: Correct

a/6 + 4 = 3

- 4 - 4

But we're guessing this last step, you may have made an error:

Step 2: Incorrect

a/6 = -1

-6 -6

✘ 22

Your approach is great,

We're guessing you may have made an error on this first step:

Step 1: Incorrect: Remember to multiply 4 by 6 as well.

6 * a/6 + 4 = 3 * 6

✘ -1/6

These problems can be tough,

We're guessing you did this first step correctly:

Step 1: Correct

c/6 + 4 = 3

- 4 - 4

But we're guessing this last step, you may have made an error:

Step 2: Incorrect: Make sure to multiply instead of dividing both sides by 6.

c/6 = -1

÷6 ÷6