Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

1. *Prove* $\overline{A \cup B} = \overline{A} \cap \overline{B}$ using builder notation:

A ∪ B = { x | x ∉ A ∪ B)}	Definition of complement
= { x | ¬ (x ∈ A ∪ B)}	Definition of ∉ ("not in")
= { x | ¬ (x ∈ A ∨ $x \in B$)}	Definition of Union
= { x | ¬ (x ∈ A) ∧ ¬ $(x \in B)$}	DeMorgan's Law
= { x | (x∉A ∧ x∉B)}	Definition of ∉
= { x | (x∈$\overline{A}$ ∧ x∈$\overline{B}$)}	Definition of complement
= { x | (x ∈ $\overline{A}$ ∩ x∈$\overline{B}$)}	Definition of intersect
= $\overline{A} \cap \overline{B}$	

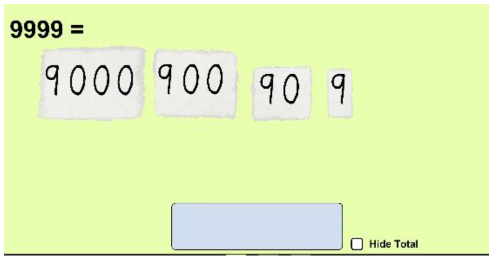
QED.

Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

2. “Make a Ten” Arithmetic Application

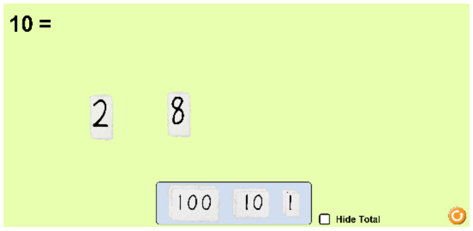
- Describe set of numbers used in “Make a Ten” application using set builder notation

  {x | x ∈ N, 0 <= x <= 9999} ; Technically, the set contains zero if no card is placed. Otherwise the set could be denoted as {x | x ∈ N, 1 <= x <= 9999}. After placing all usable cards on the screen (image below), the max value of x for all natural numbers in this “Make a Ten” application is 9999.

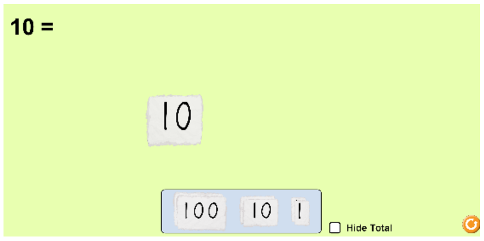


- Fundamental properties the application demonstrates (with examples)

  Closure



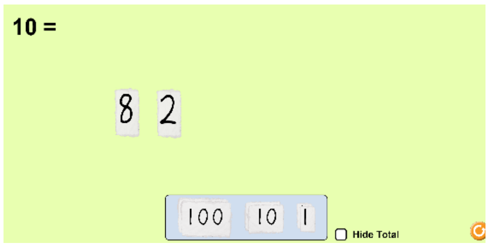
when dragging the “2” card to an “8” card, the card becomes “10” due to the closure property.



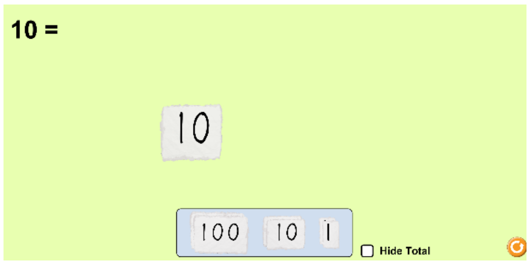
Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

Commutative

When flipping the “2” and “8”

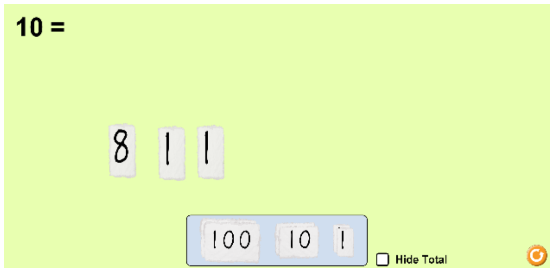


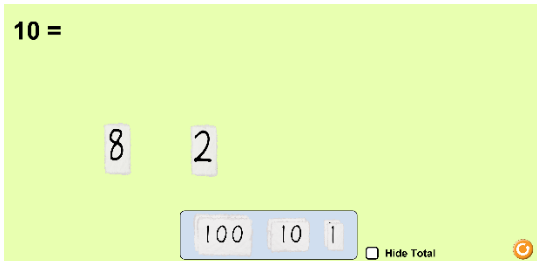
the answer still remains “10”



Associative

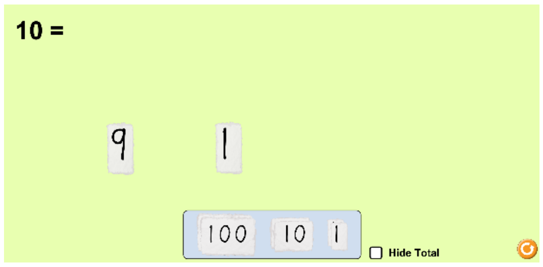
We can either add the two “1’s” first to get “8” and “2”





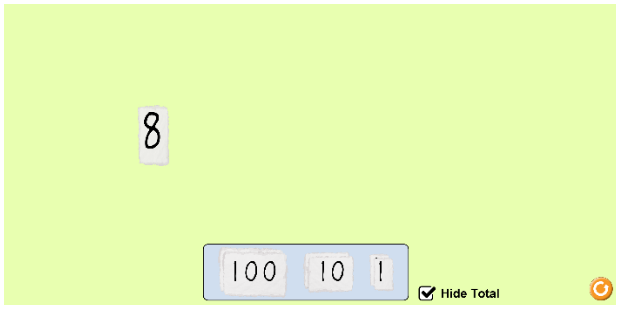
Or add “8” and “1” first to get “9” and “1”

Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

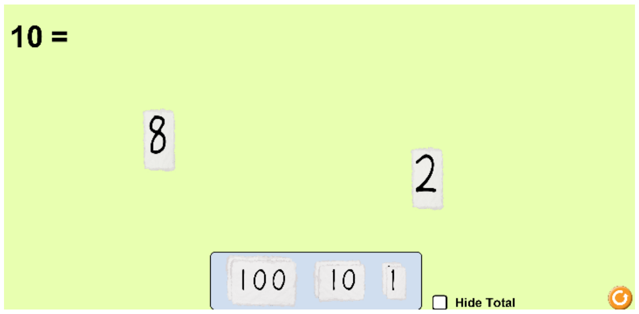


Identity

Say we have an “8.” What number exists to reach an outcome of “10”

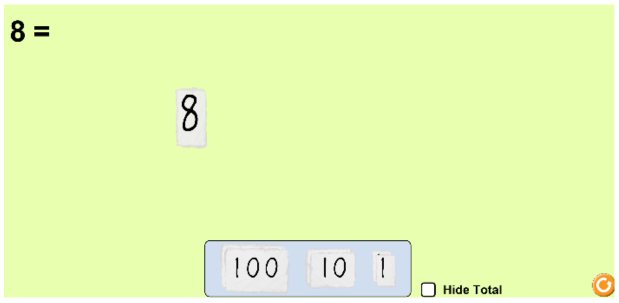


We know “2” will always yield a result of “10”



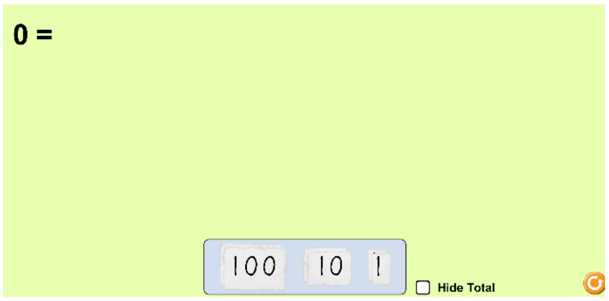
Inverse

Negative numbers weren’t used in the program but if we started with an “8”



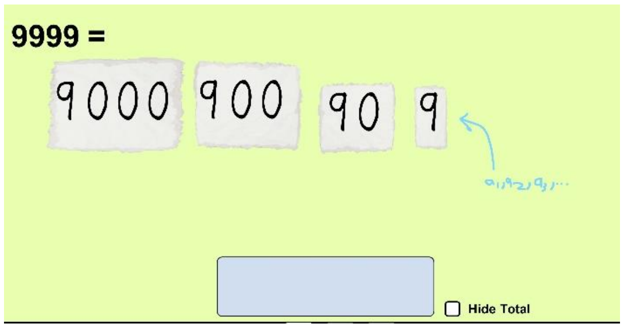
we know to get a solution of “0” we could drag the “8” away

Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms



Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

Algorithm for when to “add” two numbers



If we let the single digits placeholder to be values (a_1, a_2, a_3, etc.) then we can define an algorithm which states if the two values of the 1’s digit placeholders (a1 and a2) are added together and their sum is less than 10, combine and print the sum of $n_1 + n_2$ where n_1 can be any number from $0 <= x <= 9999$ containing a_1 value and n_2 is any number $0 <= x <= 9999$ containing a_2 value.

```
procedure sum ($a_1$, $a_2$: integers)
    # max (sum): <= 10
#For $a_1$, $a_2$ values 1 thru 9 (singles digits)
for i: 1 to 9
  #definition of function adding $a_1$ and $a_2$
  def single_digits($a_1$_$a_2$)
  sum_a1a2 = ($a_1$ + $a_2$)

  #While the sum is less than 10, add and return their sum
  while single_digits:
    if sum_a1a2 <= 10:   #the max sum of $a_1$ and $a_2$ (1's digits placeholder)
      return max_sum #combine the numbers
    #If sum is greater than 10, do not print sum or “bounce” numbers
      else: reject  #“Bounce Numbers” whenever sum_$a_1a_2$ > 10

# Values of $a_1$ and $a_2$
```

Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

```
a1 = int(input("what is 1st value of single digits placeholder: "))

a2 = int(input("what is 2nd value of single digits placeholder: "))


#calling function above with a1 and a2 values

a1_a2 = single_digits (a1, a2)


#n values (i.e. 549, 27 where 9 and 7 are the a1 and a2 values)

n1 = #value of card minus singles digits placeholder

n2 = #value of card minus singles digits placeholder


#adding n1 and n2

sum_n = (n1 + n2)


#taking sum of n values and adding to any returned sum values from function above to get final sum

#i.e. (If 549, 27 are example values, 540 + 20 =  sum_n and 9 + 7 = 16; In this case, the numbers would
"bounce" because 9 + 7 > 10. If 543 and 22 were example values, 540 + 20 = sum_n and 3 + 2 = 5, which
would then correspond to sum_a1a2.  Total would be 565 which could then display on placecards).

print sum(return(max_sum) + sum_n)
```

Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

3. Prove $b \log_{10} a = \log_{10} (a^b)$

We know,

$10^a = b$
$\log_{10} b = a$

$(10^a)^{c\,=}(b)^c$	Raise both sides by some exponent c
$10^{a*c} = b^c$	Exponent Rules
$\log_{10} (b^c)^{=} a * c$	Definition of logs
$\log_{10} (b^c)^{=} c * a$	Commutative Property
$\log_{10} (b^c)^{=} c * \log_{10} b$	Substitution

QED.

Is $b \log_{10} a = (\log_{10} a)^b$

We can take a very similar approach to show $b \log_{10} a = (\log_{10} a)^b$

$10^a = b$

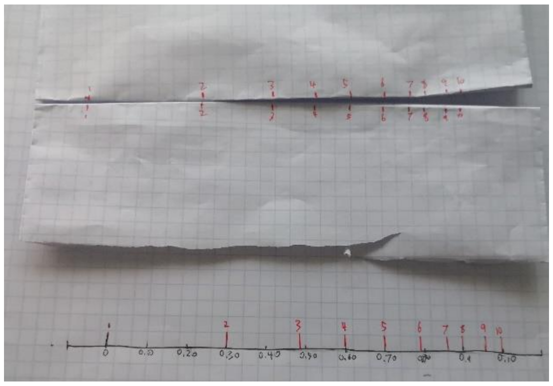
$(\log_{10(} a))^b = \log_{10} a^b$	Exponent power rule (the ^{b} outside the parenthesis can be added)
$\log_{10} b = a$	definition of log from problem 3(a)

$(10^a)^{c\,=}(b)^c$	Raise both sides by some exponent c
$10^{a*c} = b^c$	Exponent Rules
$\log_{10} (b^c)^{=} a * c$	Definition of logs
$\log_{10} (b^c)^{=} c * a$	Commutative Property
$\log_{10} (b^c)^{=} c * \log_{10} b$	Substitution

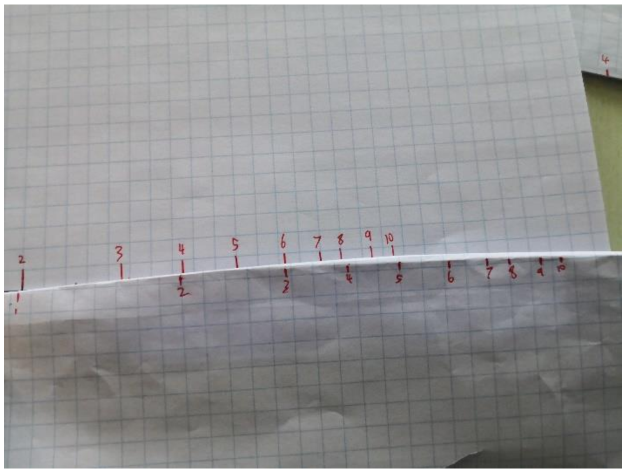
QED.

Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

4.

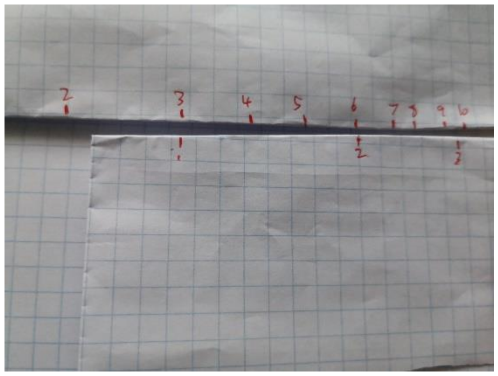


a. To calculate 2 x 3 using the slide rule, you can simply move the bottom slide to the right until the “1” matches up with the “2” of the upper slide (see image below)



Now that the “2” corresponds with the “1,” we just look to see which number is corresponding the “3” from the bottom slide to calculate 2 x 3. Looking at the above image, the “3” matches up with the “6” so the slide rule is verified.

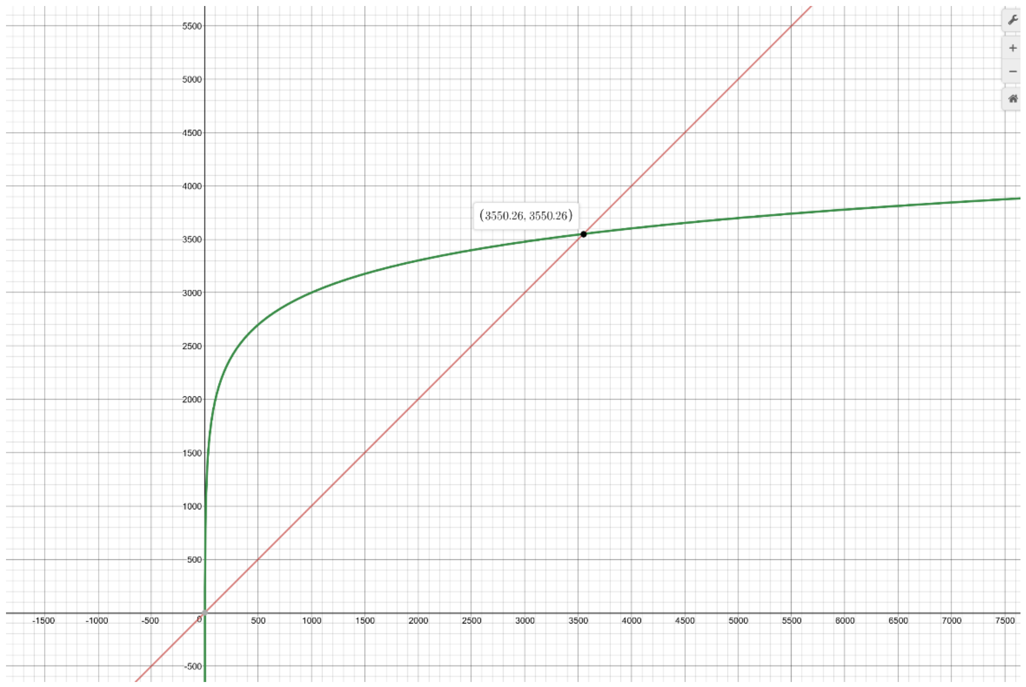
b. In order to divide 2 numbers, we can take the opposite approach. Let's say we wanted to calculate 6 ÷ 2. We could position the "2" from the bottom slide so it matches with the "6" from the upper slide (see image)



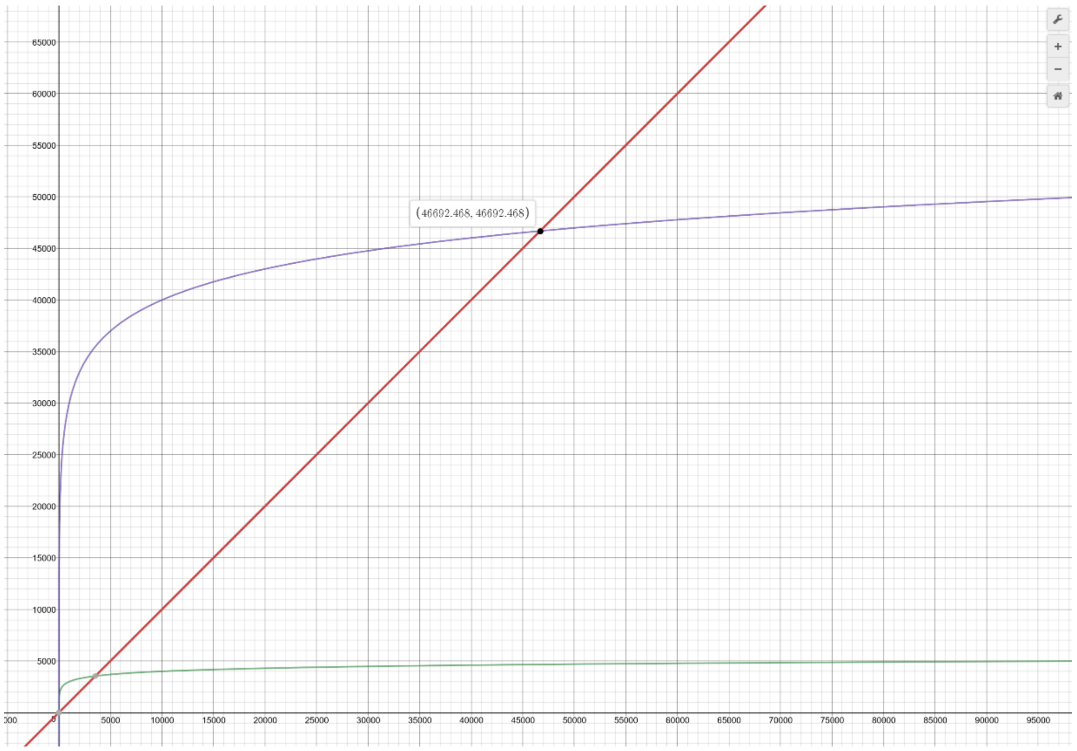
With the "6" and "2" matched up, we just have to check which number corresponds with the "1" from the bottom slide. In this case, the answer is 3. If we wanted to check a calculation a bit more complex, we could see what 4 ÷ 3 would correlate with using the slide rule. After matching the "4" from the top slide with "3" with the bottom slide, the "1" doesn't directly fall on a number, however, given it's approximately 1/3 of the distance between "1" and "2" on the top slide, we can approximate the value of 4 ÷ 3 to be 1.3 (image unavailable).

Theo Shin
Problem Set 5 - Sets, Functions, and Algorithms

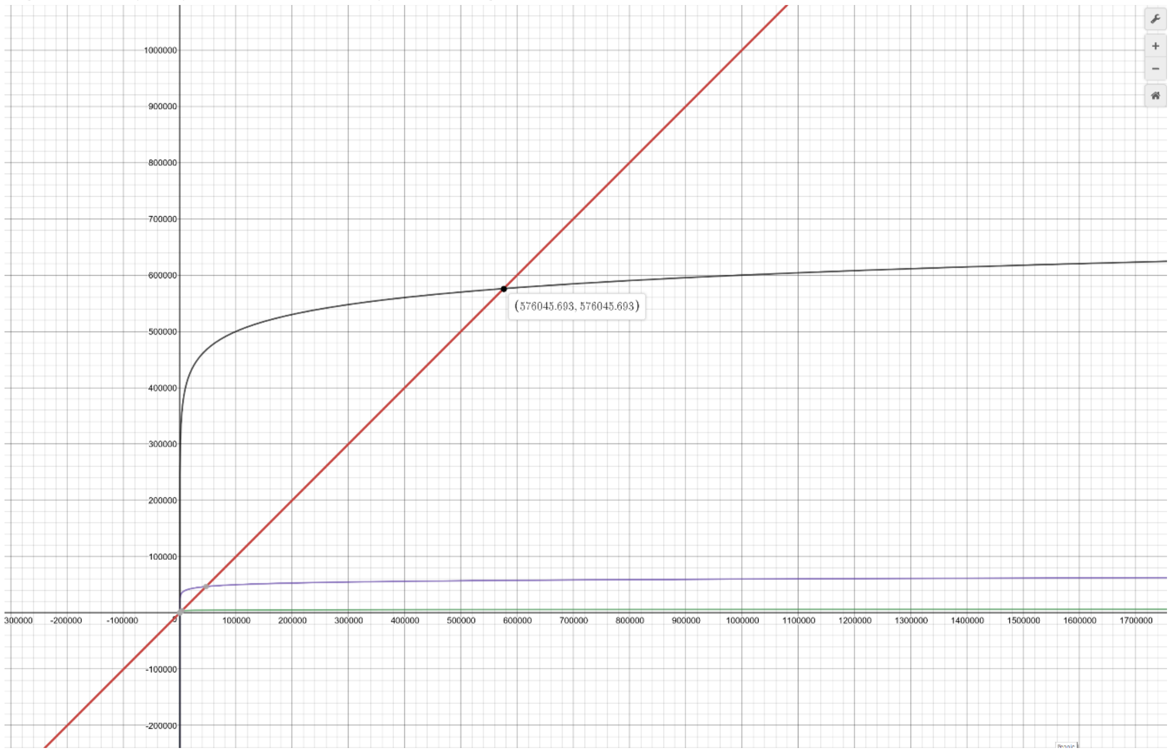
5. From initial glance and assuming unfamiliarity with the graphical representation of log functions, it appears g(x) is vertical and will always "win" compared to y = x. By significantly panning out, it appears f(x) intersects with g(x) at x,y coordinate (3550.26, 3550.26).



Similarly, for p(x) = 10,000 log (x), we have to pan out even more to see f(x) meets and "surpasses" p(x) at coordinate (46692.468, 46692.468). We can observe the magnitude of difference between g(x) and p(x) by comparing the purple and green lines.



For t(x) = 100,000 log (x) we can see the intersection occurs at (576045.693, 576045.693), significantly expanded out compared to g(x) and t(x).



- Can we ever get a log function to "beat" y = x?

  Though unnecessary, I tested 1,000,000 log (x). The intersection of f(x) and this new line occurred at (1 x $10^{10,}$ 1 x 10^{10}). It's apparent no matter how large the log function gets, it will always intersect with y = x. Because log functions are inverses of exponential functions, the graphical representation of log vs exponents is mirrored along the line formed by y = x. So, in contrast, the graph of exponents functions will be "beat" by y=x up until a certain point where it will always be greater than y=x.