Before we start

If you or someone you know is suffering from food and/or housing insecurities there are UCSD resources here to help:

Basic Needs Office: https://basicneeds.ucsd.edu/

Triton Food Pantry (in the old Student Center) is free and anonymous, and includes produce:

https://www.facebook.com/tritonfoodpantry/

Mutual Aid UCSD: https://mutualaiducsd.wordpress.com/

If you find yourself in an uncomfortable situation, ask for help. We are committed to upholding University policies regarding nondiscrimination, sexual violence and sexual harassment.

Counseling and Psychological Services (CAPS) at 858 5343755 or http://caps.ucsd.edu

OPHD at (858) 534-8298, ophd@ucsd.edu , http://ophd.ucsd.edu. CARE at Sexual Assault Resource Center at 858 5345793 sarc@ucsd.edu http://care.ucsd.edu

Pandemic resilient instruction

Fall 2021 is a transition quarter so please be patient with us as we do our best to serve the needs of all students while adhering to the university guidelines. First and foremost is the health and safety of everyone. Please do not come to class if you are sick or even think you might be sick. Please reach out (minnes@eng.ucsd.edu) if you need support with extenuating circumstances.

Masks are required in class. All students who attend class must also be fully vaccinated against COVID-19 unless they have a university-approved exemption. Campus policy requires masks and daily "symptom screeners" for everyone and we expect all students to follow these rules.

Welcome to CSE 20: Discrete Math for Computer Science in Fall 2021!

Themes and applications for CSE 20

- **Technical skepticism**: Know, select and apply appropriate computing knowledge and problem-solving techniques. Reason about computation and systems. Use mathematical techniques to solve problems. Determine appropriate conceptual tools to apply to new situations. Know when tools do not apply and try different approaches. Critically analyze and evaluate candidate solutions.

- **Multiple representations**: Understand, guide, shape impact of computing on society/the world. Connect the role of Theory CS classes to other applications (in undergraduate CS curriculum and beyond). Model problems using appropriate mathematical concepts. Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.

Applications: Numbers (how to represent them and use them in Computer Science), Recommendation systems and their roots in machine learning (with applications like Netflix), "Under the hood" of computers (circuits, pixel color representation, data structures), Codes and information (secret message sharing and error correction), Bioinformatics algorithms and genomics (DNA and RNA).

Introductions

Class website: http://cseweb.ucsd.edu/classes/fa21/cse20-a

Pro-tip: the URL structure is your map to finding your course website for other CSE classes.

Pro-tip: you can use MATH109 to replace CSE20 for prerequisites and other requirements.

Instructor: Prof. Mia Minnes "Minnes" rhymes with Guinness, minnes@eng.ucsd.edu, http://cseweb.ucsd.edu/ minnes

Our team: Four TAs and 10 tutors + all of you

Fill in contact info for students around you, if you'd like:

On a typical week: **MWF** Lectures + review quizzes, **T** HW due, **W** Discussion, office hours, Piazza. Project parts will be due some weeks.

All dates are on Canvas (click for link) and details are on course calendar (click for link).

Education research: CSE 20 is participating in a project on retention and sense of community in UCSD majors; see research plan. If you consent to participate in this study, no action is needed. If you DO NOT consent to participate in this study, or you choose to opt-out at any time during the a cademic year, sign and submit this form to the research contact at retentionstudy@cs.ucsd.edu.

Friday September 24

What data should we encode about each Netflix account holder to help us make effective recommendations?

In machine learning, clustering can be used to group similar data for prediction and recommendation. For example, each Netflix user's viewing history can be represented as a n-tuple indicating their preferences about movies in the database, where n is the number of movies in the database. People with similar tastes in movies can then be clustered to provide recommendations of movies for one another. Mathematically, clustering is based on a notion of distance between pairs of n-tuples.

In the table below, each row represents a user's ratings of movies: ✓ (check) indicates the person liked the movie, ✗ (x) that they didn't, and ● (dot) that they didn't rate it one way or another (neutral rating or didn't watch). Can encode these ratings numerically with 1 for ✓ (check), -1 for ✗ (x), and 0 for ● (dot).

Person	Fyre	Frozen II	Picard	Ratings written as a 3-tuple
P_1	✗	●	✓	
P_2	✓	✓	✗	
P_3	✓	✓	✓	
P_4	●	✗	✓	

Conclusion: Modeling involves choosing data types to represent and organize data

Review: Week 0 Friday

1. Please complete the beginning of the quarter survey https://forms.gle/gvibFnNixxqcWbaU8

2. We want you to be familiar with class policies and procedures so you are ready to have a successful quarter. Please take a look at the class website http://cseweb.ucsd.edu/classes/fa21/cse20-a and answer the questions about it on Gradescope.

3. Modeling:

   (a) Using the example movie database from class with the 3 movies Fyre, Frozen II, Picard, which of the following is a 3-tuple that represents the ratings of a user who liked Frozen II? (Select all and only that apply.)

      i. 1
      ii. $(0, 0, 0)$
      iii. $[1, 1, 1]$
      iv. $\{-1, 0, 1\}$
      v. $(1, -1, 0)$
      vi. $(0, 1, 1)$
      vii. $(1, 1, 1, 1)$

   (b) Using the example movie database from class with the 3 movies Fyre, Frozen II, Picard, how many distinct (different) 3-tuples of ratings are there?

Monday September 27

Notation and prerequisites

Term	Notation Example(s)	We say in English ...
sequence	$x_1, \ldots, x_n$	A sequence x_1 to x_n
summation	$\sum_{i=1}^{n} x_i$ or $\displaystyle\sum_{i=1}^{n} x_i$	The sum of the terms of the sequence x_1 to x_n
all reals	$\mathbb{R}$	The (set of all) real numbers (numbers on the number line)
all integers	$\mathbb{Z}$	The (set of all) integers (whole numbers including negatives, zero, and positives)
all positive integers	$\mathbb{Z}^+$	The (set of all) strictly positive integers
all natural numbers	$\mathbb{N}$	The (set of all) natural numbers. **Note**: we use the convention that 0 is a natural number.
piecewise rule definition	$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$	Define f of x to be x when x is nonnegative and to be $-x$ when x is negative
function application	$f(7)$ $f(z)$ $f(g(z))$	f of 7 **or** f applied to 7 **or** the image of 7 under f f of z **or** f applied to z **or** the image of z under f f of g of z **or** f applied to the result of g applied to z
absolute value	$\lvert -3 \rvert$	The absolute value of -3
square root	$\sqrt{9}$	The non-negative square root of 9

Data Types: sets, n-tuples, and strings

Term	**Examples**: (add additional examples from class)
set unordered collection of elements *repetition doesn't matter* *Equal sets agree on membership of all elements*	$7 \in \{43, 7, 9\}$ $2 \notin \{43, 7, 9\}$
n-tuple ordered sequence of elements with n "slots" $(n > 0)$ *repetition matters, fixed length* *Equal n-tuples have corresponding components equal*	
string ordered finite sequence of elements each from specified set *repetition matters, arbitrary finite length* *Equal strings have same length and corresponding characters equal*	

Special cases:

When $n = 2$, the 2-tuple is called an **ordered pair**.

A string of length 0 is called the **empty string** and is denoted λ.

A set with no elements is called the **empty set** and is denoted $\{\}$ or $\emptyset$.

To define sets:

To define a set using **roster method**, explicitly list its elements. That is, start with $\{$ then list elements of the set separated by commas and close with $\}$.

To define a set using **set builder definition**, either form "The set of all x from the universe U such that x is ..." by writing
$$\{x \in U \mid ...x...\}$$
or form "the collection of all outputs of some operation when the input ranges over the universe U" by writing
$$\{...x... \mid x \in U\}$$

We use the symbol $\in$ as "is an element of" to indicate membership in a set.

Example sets: For each of the following, identify whether it's defined using the roster method or set builder notation and give an example element.

$\{-1, 1\}$

$\{0, 0\}$

$\{-1, 0, 1\}$

$\{(x, x, x) \mid x \in \{-1, 0, 1\}\}$

$\{\}$

$\{x \in \mathbb{Z} \mid x \geq 0\}$

$\{x \in \mathbb{Z} \mid x > 0\}$

$\{\texttt{A}, \texttt{C}, \texttt{U}, \texttt{G}\}$

$\{\texttt{AUG}, \texttt{UAG}, \texttt{UGA}, \texttt{UAA}\}$

RNA is made up of strands of four different bases that encode genomic information in specific ways. The bases are elements of the set $B = \{\texttt{A}, \texttt{C}, \texttt{U}, \texttt{G}\}$.

Formally, to define the set of all RNA strands, we need more than roster method or set builder descriptions.

New! Recursive Definitions of Sets: The set S (pick a name) is defined by:

Basis Step:	Specify finitely many elements of S
Recursive Step:	Give rule(s) for creating a new element of S from known values existing in S, and potentially other values.

The set S then consists of all and only elements that are put in S by finitely many (a nonnegative integer number) of applications of the recursive step after the basis step.

Definition The set of nonnegative integers $\mathbb{N}$ is defined (recursively) by:

Basis Step:
Recursive Step:

Examples:

Definition The set of all integers $\mathbb{Z}$ is defined (recursively) by:

Basis Step:
Recursive Step:

Examples:

Definition The set of RNA strands S is defined (recursively) by:

Basis Step:	$\texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

Examples:

Definition The set of bitstrings (strings of 0s and 1s) is defined (recursively) by:

Basis Step:
Recursive Step:

Notation: We call the set of bitstrings $\{0, 1\}^*$.

Examples:

Review: Week 1 Monday

1. Colors can be described as amounts of red, green, and blue mixed together[1] Mathematically, a color can be represented as a 3-tuple (r, g, b) where r represents the red component, g the green component, b the blue component and where each of r, g, b must be a value from this collection of numbers:

   {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255}

   (a) **True** or **False**: $(1, 3, 4)$ fits the definition of a color above.

   (b) **True** or **False**: $(1, 100, 200, 0)$ fits the definition of a color above.

   (c) **True** or **False**: $(510, 255)$ fits the definition of a color above.

   (d) **True** or **False**: There is a color (r_1, g_1, b_1) where $r_1 + g_1 + b_1$ is greater than 765.

   (e) **True** or **False**: There is a color (r_2, g_2, b_2) where $r_2 + g_2 + b_2$ is equal to 1.

   (f) **True** or **False**: Another way to write the collection of allowed values for red, green, and blue components is
   $$\{x \in \mathbb{N} \mid 0 \leq x \leq 255\}$$
   .

   (g) **True** or **False**: Another way to write the collection of allowed values for red, green, and blue components is
   $$\{n \in \mathbb{Z} \mid 0 \leq n \leq 255\}$$
   .

   (h) **True** or **False**: Another way to write the collection of allowed values for red, green, and blue components is
   $$\{y \in \mathbb{Z} \mid -1 < y \leq 255\}$$
   .

2. Sets are unordered collections. In class, we saw some examples of sets and also how to define sets using roster method and set builder notation.

   (a) Select all and only the sets below that have 0 as an element.

   　　i. $\{-1, 1\}$

---

[1]This RGB representation is common in web applications. Many online tools are available to play around with mixing these colors, e.g. `https://www.w3schools.com/colors/colors_rgb.asp`.

ii. $\{0, 0\}$

iii. $\{-1, 0, 1\}$

iv. $\mathbb{Z}$

v. $\mathbb{Z}^+$

vi. $\mathbb{N}$

(b) Select all and only the sets below that have the ordered pair $(2, 0)$ as an element.

i. $\{x \mid x \in \mathbb{N}\}$

ii. $\{(x, x) \mid x \in \mathbb{N}\}$

iii. $\{(x, x - 2) \mid x \in \mathbb{N}\}$

iv. $\{(x, y) \mid x \in \mathbb{Z}^+, y \in \mathbb{Z}\}$

3. Which of the following are (recursive) definitions of the set of integers $\mathbb{Z}$? (Select True/False for each one.)

(a)

Basis Step: $5 \in \mathbb{Z}$

Recursive Step: If $x \in \mathbb{Z}$, then $x + 1 \in \mathbb{Z}$ and $x - 1 \in \mathbb{Z}$

(b)

Basis Step: $0 \in \mathbb{Z}$

Recursive Step: If $x \in \mathbb{Z}$, then $x + 1 \in \mathbb{Z}$ and $x - 1 \in \mathbb{Z}$ and $x + 2 \in \mathbb{Z}$ and $x - 2 \in \mathbb{Z}$

(c)

Basis Step: $0 \in \mathbb{Z}$

Recursive Step: If $x \in \mathbb{Z}$, then $x + 2 \in \mathbb{Z}$ and $x - 1 \in \mathbb{Z}$

(d)

Basis Step: $0 \in \mathbb{Z}$

Recursive Step: If $x \in \mathbb{Z}$, then $x + 1 \in \mathbb{Z}$ and $x + 2 \in \mathbb{Z}$

To define a set we can use the roster method, set builder notation, a recursive definition, and also we can apply a set operation to other sets.

New! Cartesian product of sets and **set-wise concatenation of sets of strings**

Definition: Let X and Y be sets. The **Cartesian product** of X and Y, denoted $X \times Y$, is the set of all ordered pairs (x, y) where $x \in X$ and $y \in Y$

$$X \times Y = \{(x, y) \mid x \in X \text{ and } y \in Y\}$$

Definition: Let X and Y be sets of strings over the same alphabet. The **set-wise concatenation** of X and Y, denoted $X \circ Y$, is the set of all results of string concatenation xy where $x \in X$ and $y \in Y$

$$X \circ Y = \{xy \mid x \in X \text{ and } y \in Y\}$$

Pro-tip: the meaning of writing one element next to another like xy depends on the data-types of x and y. When x and y are strings, the convention is that xy is the result of string concatenation. When x and y are numbers, the convention is that xy is the result of multiplication. This is (one of the many reasons) why is it very important to declare the data-type of variables before we use them.

Fill in the missing entries in the table:

Set	Example elements in this set:
B	`A` `C` `G` `U`
	$(\texttt{A}, \texttt{C})$ $\quad$ $(\texttt{U}, \texttt{U})$
$B \times \{-1, 0, 1\}$	
$\{-1, 0, 1\} \times B$	
	$(0, 0, 0)$
$\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\} \circ \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$	
	`GGGG`

New! Defining functions A function is defined by its (1) domain, (2) codomain, and (3) rule assigning each element in the domain exactly one element in the codomain.

The domain and codomain are nonempty sets.
The rule can be depicted as a table, formula, or English description.
The notation is

> "Let the function FUNCTION-NAME: DOMAIN $\rightarrow$ CODOMAIN be given by
> FUNCTION-NAME(x) $= \ldots$ for every $x \in DOMAIN$".

or

> "Consider the function FUNCTION-NAME: DOMAIN $\rightarrow$ CODOMAIN given by
> FUNCTION-NAME(x) $= \ldots$ for every $x \in DOMAIN$".

Example: The absolute value function

Domain

Codomain

Rule

Recall our representation of Netflix users' ratings of movies as n-tuples, where n is the number of movies in the database. Each component of the n-tuple is -1 (didn't like the movie), 0 (neutral rating or didn't watch the movie), or 1 (liked the movie).

Consider the ratings $P_1 = (-1, 0, 1)$, $P_2 = (1, 1, -1)$, $P_3 = (1, 1, 1)$, $P_4 = (0, -1, 1)$

Which of P_1, P_2, P_3 has movie preferences most similar to P_4?

One approach to answer this question: use **functions** to define distance between user preferences.

For example, consider the function d_0 : $\rightarrow$
given by
$$d_0(\, (\, (x_1, x_2, x_3), (y_1, y_2, y_3)\,)\,) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

Extra example: A new movie is released, and P_1 and P_2 watch it before P_3, and give it ratings; P_1 gives ✓ and P_2 gives ✗. Should this movie be recommended to P_3? Why or why not?

Extra example: Define a new function that could be used to compare the 4-tuples of ratings encoding movie preferences now that there are four movies in the database.

When the domain of a function is a *recursively defined set*, the rule assigning images to domain elements (outputs) can also be defined recursively.

Recall: The set of RNA strands S is defined (recursively) by:

$$\text{Basis Step:} \quad \mathtt{A} \in S, \mathtt{C} \in S, \mathtt{U} \in S, \mathtt{G} \in S$$
$$\text{Recursive Step:} \quad \text{If } s \in S \text{ and } b \in B, \text{ then } sb \in S$$

where sb is string concatenation.

Definition (Of a function, recursively) A function *rnalen* that computes the length of RNA strands in S is defined by:

$$
\begin{array}{lll}
 & & rnalen : S \to \mathbb{Z}^+ \\
\text{Basis Step:} & \text{If } b \in B \text{ then} & rnalen(b) = 1 \\
\text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B, \text{ then} & rnalen(sb) = 1 + rnalen(s)
\end{array}
$$

The domain of *rnalen* is

The codomain of *rnalen* is

Example function application:

$$rnalen(\mathtt{ACU}) =$$

Extra example: A function *basecount* that computes the number of a given base b appearing in a RNA strand s is defined recursively:

$$
\begin{array}{lll}
 & & basecount : S \times B \to \mathbb{N} \\
\text{Basis Step:} & \text{If } b_1 \in B, b_2 \in B & basecount(\,(b_1, b_2)\,) = \begin{cases} 1 & \text{when } b_1 = b_2 \\ 0 & \text{when } b_1 \neq b_2 \end{cases} \\
\text{Recursive Step:} & \text{If } s \in S, b_1 \in B, b_2 \in B & basecount(\,(sb_1, b_2)\,) = \begin{cases} 1 + basecount(\,(s, b_2)\,) & \text{when } b_1 = b_2 \\ basecount(\,(s, b_2)\,) & \text{when } b_1 \neq b_2 \end{cases}
\end{array}
$$

$$basecount(\,(\mathtt{ACU}, \mathtt{A})\,) = basecount(\,(\mathtt{AC}, \mathtt{A})\,) = basecount(\,(\mathtt{A}, \mathtt{A})\,) = 1$$

$$basecount(\,(\mathtt{ACU}, \mathtt{G})\,) = basecount(\,(\mathtt{AC}, \mathtt{G})\,) = basecount(\,(\mathtt{A}, \mathtt{G})\,) = 0$$

Extra example: The function which outputs 2^n when given a nonnegative integer n can be defined recursively, because its domain is the set of nonnegative integers.

Review: Week 1 Wednesday

1. RNA is made up of strands of four different bases that encode genomic information in specific ways. The bases are elements of the set $B = \{\text{A}, \text{C}, \text{G}, \text{U}\}$. The set of RNA strands S is defined (recursively) by:

   Basis Step: $\text{A} \in S, \text{C} \in S, \text{U} \in S, \text{G} \in S$

   Recursive Step: If $s \in S$ and $b \in B$, then $sb \in S$

   A function *rnalen* that computes the length of RNA strands in S is defined by:

   $$rnalen : S \;\; \to \mathbb{Z}^+$$

   Basis Step: If $b \in B$ then $rnalen(b) \;\; = 1$

   Recursive Step: If $s \in S$ and $b \in B$, then $rnalen(sb) \;\; = 1 + rnalen(s)$

   (a) How many distinct elements are in the set described using set builder notation as
   $$\{x \in S \mid rnalen(x) = 1\} \qquad ?$$

   (b) How many distinct elements are in the set described using set builder notation as
   $$\{x \in S \mid rnalen(x) = 2\} \qquad ?$$

   (c) How many distinct elements are in the set described using set builder notation as
   $$\{rnalen(x) \mid x \in S \text{ and } rnalen(x) = 2\} \qquad ?$$

   (d) How many distinct elements are in the set obtained as the result of the set-wise concatenation $\{\text{AA}, \text{AC}\} \circ \{\text{U}, \text{AA}\}$?

   (e) How many distinct elements are in the set obtained as the result of the Cartesian product $\{\text{AA}, \text{AC}\} \times \{\text{U}, \text{AA}\}$?

   (f) **True** or **False**: There is an example of an RNA strand that is both in the set obtained as the result of the set-wise concatenation $\{\text{AA}, \text{AC}\} \circ \{\text{U}, \text{AA}\}$ and in the set obtained as the result of the Cartesian product $\{\text{AA}, \text{AC}\} \times \{\text{UA}, \text{AA}\}$

   Bonus - not for credit: Describe each of the sets above using roster method.

2. Recall the function d_0 which takes an ordered pair of ratings 3-tuples and returns a measure of the distance between them given by
   $$d_0(\, (\, (x_1, x_2, x_3), (y_1, y_2, y_3)\,)\,) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$
   Consider the function application
   $$d_0(\, (\, (-1, 1, 1), (1, 0, -1)\,)\,)$$

   (a) What is the input?

   (b) What is the output?

3. To give the rule for a recursive definition of the function with codomain $\mathbb{Z}$ which gives 2^n for a nonnegative integer n, fill in each step below.

   (a) Basis Step: $2^0 = \underline{\quad}$

   (b) Recursive Step: If $n \in \mathbb{N}$, then

        i. $2^n = 2^n$

        ii. $2^{n+1} = n + 2$

        iii. $2^{n+1} = 2n$

Friday October 1 (Zoom)

Modeling uses data-types that are encoded in a computer.

The details of the encoding impact the efficiency of algorithms we use to understand the systems we are modeling and the impacts of these algorithms on the people using the systems.

Case study: how to encode numbers?

Definition For b an integer greater than 1 and n a positive integer, the **base b expansion of n** is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are nonnegative integers less than b, $a_{k-1} \neq 0$, and

$$n = \sum_{i=0}^{k-1} a_i b^i$$

Notice: *The base b expansion of a positive integer n is a string over the alphabet $\{x \in \mathbb{N} \mid x < b\}$ whose leftmost character is nonzero.*

Base b	Collection of possible coefficients in base b expansion of a positive integer
Binary ($b = 2$)	$\{0, 1\}$
Ternary ($b = 3$)	$\{0, 1, 2\}$
Octal ($b = 8$)	$\{0, 1, 2, 3, 4, 5, 6, 7\}$
Decimal ($b = 10$)	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
Hexadecimal ($b = 16$)	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ letter coefficient symbols represent numerical values $(A)_{16} = (10)_{10}$ $(B)_{16} = (11)_{10}$ $(C)_{16} = (12)_{10}$ $(D)_{16} = (13)_{10}$ $(E)_{16} = (14)_{10}$ $(F)_{16} = (15)_{10}$

Examples:

$(1401)_2$

$(1401)_{10}$

$(1401)_{16}$

New! An algorithm is a finite sequence of precise instructions for solving a problem.

Algorithm for calculating integer part of half the input

```
1    procedure half(n: a positive integer)
2    r := 0
3    while n > 1
4        r := r + 1
5        n := n - 2
6    return r {r holds the result of the operation}
```

n	r	$n > 1$?
6		

n	r	$n > 1$?
5		

Algorithm for calculating integer part of log

```
1    procedure log(n: a positive integer)
2    r := 0
3    while n > 1
4        r := r + 1
5        n := half(n)
6    return r {r holds the result of the log operation}
```

n	r	$n > 1$?
8		

n	r	$n > 1$?
6		

| $2^0 = 1$ | $2^1 = 2$ | $2^2 = 4$ | $2^3 = 8$ | $2^4 = 16$ | $2^5 = 32$ | $2^6 = 64$ | $2^7 = 128$ | $2^8 = 256$ | $2^9 = 512$ | $2^{10} = 1024$ |

Integer division and remainders (aka The Division Algorithm) Let n be an integer and d a positive integer. There are unique integers q and r, with $0 \le r < d$, such that $n = dq + r$. In this case, d is called the divisor, n is called the dividend, q is called the quotient, and r is called the remainder. We write $q = n \textbf{ div } d$ and $r = n \textbf{ mod } d$.

Extra example: How do **div** and **mod** compare to / and % in Java and python?

Two algorithms for constructing base b expansion from decimal representation

Most significant first: Start with left-most coefficient of expansion

Calculating integer part of $\log_b$

```
1  procedure logb(n,b: positive integers with b>1)
2  r := 0
3  while n > b − 1
4    r := r + 1
5    n := n div b
6  return r  {r holds the result of the logb operation}
```

Calculating base b expansion, from left

```
1  procedure baseb1(n,b: positive integers with b>1)
2  v := n
3  k := logb(n,b) + 1
4  for i := 1 to k
5    a_{k−i} := 0
6    while v ≥ b^{k−i}
7      a_{k−i} := a_{k−i} + 1
8      v := v − b^{k−i}
9  return (a_{k−1},…,a_0){(a_{k−1}…a_0)_b is the base b expansion of n}
```

Least significant first: Start with right-most coefficient of expansion

Idea: (when $k > 1$)

$$n = a_{k-1}b^{k-1} + \cdots + a_1 b + a_0$$
$$= b(a_{k-1}b^{k-2} + \cdots + a_1) + a_0$$

so $a_0 = n \textbf{ mod } b$ and $a_{k-1}b^{k-2} + \cdots + a_1 = n \textbf{ div } b$.

Calculating base b expansion, from right

```
1  procedure baseb2(n,b: positive integers with b>1)
2  q := n
3  k := 0
4  while q ≠ 0
5    a_k := q mod b
6    q := q div b
7    k := k + 1
8  return (a_{k−1},…,a_0){(a_{k−1}…a_0)_b is the base b expansion of n}
```

Review: Week 1 Friday

1. For many applications in cryptography and random number generation, dividing very large integers efficiently is critical. Recall the definitions known as **The Division Algorithm**: Let n be an integer and d a positive integer. There are unique integers q and r, with $0 \leq r < d$, such that $n = dq + r$. In this case, d is called the divisor, n is called the dividend, q is called the quotient, and r is called the remainder. We write $q = n$ **div** d and $r = n$ **mod** d.

   One application of the Division Algorithm is in computing the integer part of the logarithm. When we discuss algorithms in this class, we will usually write them in pseudocode or English. Sometimes we will find it useful to relate the pseudocode to runnable code in a programming language. We will typically use Java or python for this.

Calculating log in pseudocode

   ```
   1  procedure log(n: a positive integer)
   2  r := 0
   3  while n > 1
   4    r := r + 1
   5    n := n div 2
   6  return r {r holds the result of the log operation}
   ```

Calculating log in Java

   ```java
   1  int log(int n) {
   2    if (n < 1) {
   3      throw new IllegalArgumentException();
   4    }
   5    int result = 0;
   6    while (n > 1) {
   7      result = result + 1;
   8      n = n / 2;
   9    }
   10   return result;
   11 }
   ```

   (a) Calculate 2021 **div** 20. *You may use a calculator if you like.*

   (b) Calculate 2021 **mod** 20. *You may use a calculator if you like.*

   (c) How many different possible values of r (results of taking n **mod** d) are there when we consider positive integer values of n and d is 20?

   (d) What is the smallest positive integer n which can be written as $16q + 7$ for q an integer?

   (e) What is the return value of $log(457)$? *You can run the Java version in order to calculate it.*

2. Give the value (using usual mathematical conventions) of each of the following base expansions.

   (a) $(10)_2$

   (b) $(10)_4$

   (c) $(17)_{16}$

   (d) $(211)_3$

   (e) $(3)_8$

Monday November 29

Recall:

A relation is an **equivalence relation** means it is reflexive, symmetric, and transitive.

An **equivalence class** of an element $a \in A$ with respect to an equivalence relation R on the set A is the set
$$\{s \in A \mid (a, s) \in R\}$$
We write $[a]_R$ for this set, which is the equivalence class of a with respect to R.

A **partition** of a set A is a set of non-empty, disjoint subsets $A_1, A_2, \cdots, A_n$ such that

$$A = \bigcup_{i=1}^{n} A_i = \{x \mid \exists i (x \in A_i)\}$$

Claim: For each $a \in U$, $[a]_E \neq \emptyset$.

Proof: Towards a _____ consider an arbitrary element a in U. We will work to show that $[a]_E \neq \emptyset$, namely that $\exists x \in [a]_E$. By definition of equivalence classes, we can rewrite this goal as

$$\exists x \in U \ (\ (a, x) \in E \ )$$

Towards a _____, consider $x = a$, an element of U by definition. By _____ of E, we know that $(a, a) \in E$ and thus the existential quantification has been proved.

Claim: For each $a \in U$, there is some $b \in U$ such that $a \in [b]_E$.

Towards a _____ consider an arbitrary element a in U. By definition of equivalence classes, we can rewrite the goal as
$$\exists b \in U \ (\ (b, a) \in E \ )$$

Towards a _____, consider $b = a$, an element of U by definition. By _____ of E, we know that $(a, a) \in E$ and thus the existential quantification has been proved.

Claim: For each $a, b \in U$, ($(a, b) \in E \ \rightarrow \ [a]_E = [b]_E$) and ($(a, b) \notin E \ \rightarrow \ [a]_E \cap [b]_E = \emptyset$)

Corollary: Given an equivalence relation E on set U, $\{[x]_E \mid x \in U\}$ is a partition of U.

Last time, we saw that partitions associated to equivalence relations were useful in the context of modular arithmetic. Today we'll look at a different application.

Recall that in a movie recommendation system, each user's ratings of movies is represented as a n-tuple (with the positive integer n being the number of movies in the database), and each component of the n-tuple is an element of the collection $\{-1, 0, 1\}$.

We call Rt_5 the set of all ratings 5-tuples.

Define $d : Rt_5 \times Rt_5 \to \mathbb{N}$ by

$$d(\ (\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ )\ ) = \sum_{i=1}^{5} |x_i - y_i|$$

Consider the following binary relations on Rt_5.

$$E_{proj} = \{(\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ ) \in Rt_5 \times Rt_5 \ \mid \ (x_1 = y_1) \wedge (x_2 = y_2) \wedge (x_3 = y_3)\}$$

Example ordered pair in E_{proj}:

Reflexive? Symmetric? Transitive? Antisymmetric?

$$E_{dist} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid \ d(\ (u, v)\ ) \leq 2\}$$

Example ordered pair in E_{dist}:

Reflexive? Symmetric? Transitive? Antisymmetric?

$$E_{circ} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid \ d(\ (\ (0,0,0,0,0)\ , u)\ ) = d(\ (\ (0,0,0,0,0), v\ )\ )\}$$

Example ordered pair in E_{circ}:

Reflexive? Symmetric? Transitive? Antisymmetric?

The partition of Rt_5 defined by _____ is

{ { (−1,−1,−1,−1,−1),(−1,−1,−1,−1,0),(−1,−1,−1,−1,1),(−1,−1,−1,0,−1),(−1,−1,−1,0,0),(−1,−1,−1,0,1),(−1,−1,−1,1,−1),(−1,−1,−1,1,0),(−1,−1,−1,1,1) },
{ (−1,−1,0,−1,−1),(−1,−1,0,−1,0),(−1,−1,0,−1,1),(−1,−1,0,0,−1),(−1,−1,0,0,0),(−1,−1,0,0,1),(−1,−1,0,1,−1),(−1,−1,0,1,0),(−1,−1,0,1,1) },
{ (−1,−1,1,−1,−1),(−1,−1,1,−1,0),(−1,−1,1,−1,1),(−1,−1,1,0,−1),(−1,−1,1,0,0),(−1,−1,1,0,1),(−1,−1,1,1,−1),(−1,−1,1,1,0),(−1,−1,1,1,1) },
{ (−1,0,−1,−1,−1),(−1,0,−1,−1,0),(−1,0,−1,−1,1),(−1,0,−1,0,−1),(−1,0,−1,0,0),(−1,0,−1,0,1),(−1,0,−1,1,−1),(−1,0,−1,1,0),(−1,0,−1,1,1) },
{ (−1,0,0,−1,−1),(−1,0,0,−1,0),(−1,0,0,−1,1),(−1,0,0,0,−1),(−1,0,0,0,0),(−1,0,0,0,1),(−1,0,0,1,−1),(−1,0,0,1,0),(−1,0,0,1,1) },
{ (−1,0,1,−1,−1),(−1,0,1,−1,0),(−1,0,1,−1,1),(−1,0,1,0,−1),(−1,0,1,0,0),(−1,0,1,0,1),(−1,0,1,1,−1),(−1,0,1,1,0),(−1,0,1,1,1) },
{ (−1,1,−1,−1,−1),(−1,1,−1,−1,0),(−1,1,−1,−1,1),(−1,1,−1,0,−1),(−1,1,−1,0,0),(−1,1,−1,0,1),(−1,1,−1,1,−1),(−1,1,−1,1,0),(−1,1,−1,1,1) },
{ (−1,1,0,−1,−1),(−1,1,0,−1,0),(−1,1,0,−1,1),(−1,1,0,0,−1),(−1,1,0,0,0),(−1,1,0,0,1),(−1,1,0,1,−1),(−1,1,0,1,0),(−1,1,0,1,−1) },
{ (−1,1,1,−1,−1),(−1,1,1,−1,0),(−1,1,1,−1,1),(−1,1,1,0,−1),(−1,1,1,0,0),(−1,1,1,0,1),(−1,1,1,1,−1),(−1,1,1,1,0),(−1,1,1,1,1) },
{ (0,−1,−1,−1,−1),(0,−1,−1,−1,0),(0,−1,−1,−1,1),(0,−1,−1,0,−1),(0,−1,−1,0,0),(0,−1,−1,0,1),(0,−1,−1,1,−1),(0,−1,−1,1,0),(0,−1,−1,1,1) },
{ (0,−1,0,−1,−1),(0,−1,0,−1,0),(0,−1,0,−1,1),(0,−1,0,0,−1),(0,−1,0,0,0),(0,−1,0,0,1),(0,−1,0,1,−1),(0,−1,0,1,0),(0,−1,0,1,1) },
{ (0,−1,1,−1,−1),(0,−1,1,−1,0),(0,−1,1,−1,1),(0,−1,1,0,−1),(0,−1,1,0,0),(0,−1,1,0,1),(0,−1,1,1,−1),(0,−1,1,1,0),(0,−1,1,1,1) },
{ (0,0,−1,−1,−1),(0,0,−1,−1,0),(0,0,−1,−1,1),(0,0,−1,0,−1),(0,0,−1,0,0),(0,0,−1,0,1),(0,0,−1,1,−1),(0,0,−1,1,0),(0,0,−1,1,1) },
{ (0,0,0,−1,−1),(0,0,0,−1,0),(0,0,0,−1,1),(0,0,0,0,−1),(0,0,0,0,0),(0,0,0,0,1),(0,0,0,1,−1),(0,0,0,1,0),(0,0,0,1,1) },
{ (0,0,1,−1,−1),(0,0,1,−1,0),(0,0,1,−1,1),(0,0,1,0,−1),(0,0,1,0,0),(0,0,1,0,1),(0,0,1,1,−1),(0,0,1,1,0),(0,0,1,1,1) },
{ (0,1,−1,−1,−1),(0,1,−1,−1,0),(0,1,−1,−1,1),(0,1,−1,0,−1),(0,1,−1,0,0),(0,1,−1,0,1),(0,1,−1,1,−1),(0,1,−1,1,0),(0,1,−1,1,1) },
{ (0,1,0,−1,−1),(0,1,0,−1,0),(0,1,0,−1,1),(0,1,0,0,−1),(0,1,0,0,0),(0,1,0,0,1),(0,1,0,1,−1),(0,1,0,1,0),(0,1,0,1,−1) },
{ (0,1,1,−1,−1),(0,1,1,−1,0),(0,1,1,−1,1),(0,1,1,0,−1),(0,1,1,0,0),(0,1,1,0,1),(0,1,1,1,−1),(0,1,1,1,0),(0,1,1,1,1) },
{ (1,−1,−1,−1,−1),(1,−1,−1,−1,0),(1,−1,−1,−1,1),(1,−1,−1,0,−1),(1,−1,−1,0,0),(1,−1,−1,0,1),(1,−1,−1,1,−1),(1,−1,−1,1,0),(1,−1,−1,1,1) },
{ (1,−1,0,−1,−1),(1,−1,0,−1,0),(1,−1,0,−1,1),(1,−1,0,0,−1),(1,−1,0,0,0),(1,−1,0,0,1),(1,−1,0,1,−1),(1,−1,0,1,0),(1,−1,0,1,1) },
{ (1,−1,1,−1,−1),(1,−1,1,−1,0),(1,−1,1,−1,1),(1,−1,1,0,−1),(1,−1,1,0,0),(1,−1,1,0,1),(1,−1,1,1,−1),(1,−1,1,1,0),(1,−1,1,1,1) },
{ (1,0,−1,−1,−1),(1,0,−1,−1,0),(1,0,−1,−1,1),(1,0,−1,0,−1),(1,0,−1,0,0),(1,0,−1,0,1),(1,0,−1,1,−1),(1,0,−1,1,0),(1,0,−1,1,1) },
{ (1,0,0,−1,−1),(1,0,0,−1,0),(1,0,0,−1,1),(1,0,0,0,−1),(1,0,0,0,0),(1,0,0,0,1),(1,0,0,1,−1),(1,0,0,1,0),(1,0,0,1,1) },
{ (1,0,1,−1,−1),(1,0,1,−1,0),(1,0,1,−1,1),(1,0,1,0,−1),(1,0,1,0,0),(1,0,1,0,1),(1,0,1,1,−1),(1,0,1,1,0),(1,0,1,1,1) },
{ (1,1,−1,−1,−1),(1,1,−1,−1,0),(1,1,−1,−1,1),(1,1,−1,0,−1),(1,1,−1,0,0),(1,1,−1,0,1),(1,1,−1,1,−1),(1,1,−1,1,0),(1,1,−1,1,1) },
{ (1,1,0,−1,−1),(1,1,0,−1,0),(1,1,0,−1,1),(1,1,0,0,−1),(1,1,0,0,0),(1,1,0,0,1),(1,1,0,1,−1),(1,1,0,1,0),(1,1,0,1,−1) },
{ (1,1,1,−1,−1),(1,1,1,−1,0),(1,1,1,−1,1),(1,1,1,0,−1),(1,1,1,0,0),(1,1,1,0,1),(1,1,1,1,−1),(1,1,1,1,0),(1,1,1,1,1) }
}

The partition of Rt_5 defined by $E =$ _____ is

{
    $[\ (0,0,0,0,0)\ ]_E$
   , $[\ (0,0,0,0,1)\ ]_E$
   , $[\ (0,0,0,1,1)\ ]_E$
   , $[\ (0,0,1,1,1)\ ]_E$
   , $[\ (0,1,1,1,1)\ ]_E$
   , $[\ (1,1,1,1,1)\ ]_E$
}

How many elements are in each part of the partition?

Review

1. Select all and only the partitions of $\{1, 2, 3, 4, 5\}$ from the sets below.

    (a) $\{1, 2, 3, 4, 5\}$

    (b) $\{\{1, 2, 3, 4, 5\}\}$

    (c) $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$

    (d) $\{\{1\}, \{2, 3\}, \{4\}\}$

    (e) $\{\{\emptyset, 1, 2\}, \{3, 4, 5\}\}$

2.

    Select all and only the correct statements about an equivalence relation E on a set U:

    (a) $E \in U \times U$

    (b) $E = U \times U$

    (c) $E \subseteq U \times U$

    (d) $\forall x \in U \ ([x]_E \in U)$

    (e) $\forall x \in U \ ([x]_E \subseteq U)$

    (f) $\forall x \in U \ ([x]_E \in \mathcal{P}(U))$

    (g) $\forall x \in U \ ([x]_E \subseteq \mathcal{P}(U))$

Wednesday December 1

Scenario: Good morning! You're a user experience engineer at Netflix. A product goal is to design customized home pages for groups of users who have similar interests. Your manager tasks you with designing an algorithm for producing a clustering of users based on their movie interests, so that customized homepages can be engineered for each group.

Conventions for today: We will use $U = \{r_1, r_2, \cdots, r_t\}$ to refer to an arbitrary set of user ratings (we'll pick some specific examples to explore) that are a subset of Rt_5. We will be interested in creating partitions $C_1, \cdots, C_m$ of U. We'll assume that each user represented by an element of U has a unique ratings tuple.

Your idea: equivalence relations! You offer your manager three great options:

$$E_{id} = \{(\,(x_1, x_2, x_3, x_4, x_5), (x_1, x_2, x_3, x_4, x_5)\,) \mid (x_1, x_2, x_3, x_4, x_5) \in Rt_5\}$$

Describe how each homepage should be designed . . .

$$E_{proj} = \{(\,(x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\,) \in Rt_5 \times Rt_5 \mid (x_1 = y_1) \wedge (x_2 = y_2) \wedge (x_3 = y_3)\}$$

Describe how each homepage should be designed . . .

$$E_{circ} = \{(u, v) \in Rt_5 \times Rt_5 \mid d(\,(\,(0,0,0,0,0)\,, u)\,) = d(\,(\,(0,0,0,0,0), v\,)\,)\}$$

Describe how each homepage should be designed . . .

Scenario: Good morning! You're a user experience engineer at Netflix. A product goal is to design customized home pages for groups of users who have similar interests. You task your team with designing an algorithm for producing a clustering of users based on their movie interests. Your team implements two algorithms that produce different clusterings. How do you decide which one to use? What feedback do you give the team in order to help them improve? Clearly, you will need to use math.

Your idea: find a way to **score** clusterings (partitions)

Definition: For a cluster of ratings $C = \{r_1, r_2, \cdots, r_n\} \subseteq U$, the **diameter** of the cluster is defined by:

$$diameter(C) = \max_{1 \leq i,j \leq n} \left(d(\ (r_i, r_j)\ ) \right)$$

Consider $x = (1, 0, 1, 0, 1)$, $y = (1, 1, 1, 0, 1)$, $z = (-1, -1, 0, 0, 0)$, $w = (0, 0, 0, 1, 0)$.

What is $diameter(\{x, y, z\})$? $diameter(\{x, y\})$? $diameter(\{x, z, w\})$?

diameter works on single clusters. One way to aggregate across a clustering $C_1, \cdots, C_m$ is $\sum_{k=1}^{m} diameter(C_k)$

Is this a good score?

How can we express the idea of **many elements within a small area**? Key idea: "give credit" to small diameter clusters with many elements.

Definition: For a cluster of ratings $C = \{r_1, r_2, \cdots, r_n\} \subseteq U$, the **density** of the cluster is defined by:

$$\frac{n}{1 + diameter(C)}$$

Can you use density to decide whether the partition given by the equivalence classes of E_{proj} or E_{circ} for this task?

Looking forward

Tips for future classes from the CSE 20 TAs and tutors

- In class

  - Go to class

  - Show up to class early because sometimes seats get taken/ the classroom gets full and then you have to sit on the floor

  - There's usually a space for skateboards/longboards/eboards to go at the front or rear of the lecture hall

  - If you have a flask water bottle please ensure that its secured during a lecture and it cannot fall - putting on the floor often leads to it falling since people sometimes cross your seats.

  - Take notes - it's much faster and more effective to note-take in class than watch recordings after, particularly if you do so longhand

  - Resist the urge to sit in the back. You will be able to focus much better sitting near the front, where there are fewer screens in front of you to distract from the lecture content

  - If you bring your laptop to class to take notes / access class materials, sit towards the back of the room to minimize distractions for people sitting behind you!

  - On zoom it's easy to just type a question out in chat, it might be a little more awkward to do so in person, but it is definitely worth it. Don't feel like you should already know what's being covered

  - Always check you have your iclicker[2] on you. Just keep it in your backpack permanently. That way you can never forget it.

  - Don't be afraid to talk to the people next to you during group discussions. Odds are they're as nervous as you are, and you can all benefit from sharing your thoughts and understanding of the material

  - Certain classes will podcast the lectures, just like Zoom archives lecture recordings, at podcast.ucsd.edu

  - If they aren't podcasted, and you want to record lectures, ask your professor for consent first

- Office hours, tutor hours, and the CSE building

  - Office hours are a good place to hang out and get work done while being able to ask questions as they come up

  - Office hour attendance is typically much busier in person (and confined to the space in the room)

  - Get to know the CSE building: CSE B260, basement labs, office hours rooms

  - Know how to get in to the building after-hours

- Libraries and on-campus resources

  - Look up what library floors are for what, how to book rooms: East wing of Geisel is open 24/7 (they might ask to see an ID if you stay late), East Wing of Geisel has chess boards and jigsaw puzzles, study pods on the 8th floor, free computers/wifi

---

[2]iclickers are used in many classes to encourage active participation in class. They're remotes that allow you to respond to multiple choice questions and the instructor can show a histogram of responses in real time.

- – Know Biomed exists and is usually less crowded
  - – Most libraries allow you to borrow whiteboards and markers (also laptops, tablets, microphones, and other cool stuff) for 24 hours
  - – Take advantage of Dine with a prof / Coffee with a prof program. It's legit free food / coffee once per quarter.
  - – When planning out your daily schedule, think about where classes are, how much time will they take, are their places to eat nearby and how you can schedule social time with friends to nearby areas
  - – Take into account the distances between classes if they are back to back

- Final exams

  - – What are 8am finals? Basically in-person exams are different
  - – Don't forget your university card during exams
  - – Blue books for exams (what they are, where to get them)
  - – Seating assignments for exams and go early to make sure you're in the right place (check the exits to make sure you're reading it the right way)
  - – Know where your exam is being held (find it on a map at least a day beforehand). Finals are often in strange places that take a while to find

CSE department course numbering system

Lower division

- CSE 12, Basic Data Structures and Object-Oriented Programming
- CSE 15L (2 units), Software Tools and Technique Laboratory
- CSE 20 or Math 15A, Introduction to Discrete Mathematics
- CSE 21 Mathematics for Algorithms and Systems
- CSE 30, Computer Organization & Systems Programming

Upper division

- Advanced Data Structures and Programming: CSE 100
- Theory and Algorithms: CSE 101, CSE 105
- Software Engineering: CSE 110, CSE 112
- Systems/Networks: CSE 120 or CSE 123 or CSE 124
- Programming Languages /Databases: CSE 130 or CSE 132A
- Security/Cryptography: CSE 107 or CSE 127
- AI / Machine Learning/ Vision/ Graphics: CSE 150A or CSE 151A or CSE 151B or CSE 152A or CSE 158 or CSE 167
- Hardware / Architecture: CSE 140/ CSE 140L Components and Design Techniques for Digital Systems Architecture, CSE 141 / CSE 141L Introduction to Computer Architecture and CSE 141L Project in Computer Architecture (2 units), CSE 142 / CSE 142L Comp Arch Software Perspective

Review

1.

Assume there are five movies in the database, so that each user's ratings can be represented as a 5-tuple. We call Rt_5 the set of all ratings 5-tuples. Consider the binary relation on the set of all 5-tuples where each component of the 5-tuple is an element of the collection $\{-1, 0, 1\}$

$$G_1 = \{(u, v) \mid \text{the ratings of users } u \text{ and } v \text{ agree about the first movie in the database}\}$$

$$G_2 = \{(u, v) \mid \text{the ratings of users } u \text{ and } v \text{ agree about at least two movies}\}$$

(a) **True** or **False**: The relation G_1 holds of $u = (1, 1, 1, 1, 1)$ and $v = (-1, -1, -1, -1, -1)$

(b) **True** or **False**: The relation G_2 holds of $u = (1, 0, 1, 0, -1)$ and $v = (-1, 0, 1, -1, -1)$

(c) **True** or **False**: G_1 is reflexive; namely, $\forall u \; (\; (u, u) \in G_1 \;)$

(d) **True** or **False**: G_1 is symmetric; namely, $\forall u \; \forall v \; (\; (u, v) \in G_1 \rightarrow (v, u) \in G_1 \;)$

(e) **True** or **False**: G_1 is transitive; namely, $\forall u \; \forall v \; \forall w (\; ((u, v) \in G_1 \wedge (v, w) \in G_1) \rightarrow (u, w) \in G_1 \;)$

(f) **True** or **False**: G_2 is reflexive; namely, $\forall u \; (\; (u, u) \in G_2 \;)$

(g) **True** or **False**: G_2 is symmetric; namely, $\forall u \; \forall v \; (\; (u, v) \in G_2 \rightarrow (v, u) \in G_2 \;)$

(h) **True** or **False**: G_2 is transitive; namely, $\forall u \; \forall v \; \forall w (\; ((u, v) \in G_2 \wedge (v, w) \in G_2) \rightarrow (u, w) \in G_2 \;)$

2.

Recall that in a movie recommendation system, each user's ratings of movies is represented as a n-tuple (with the positive integer n being the number of movies in the database), and each component of the n-tuple is an element of the collection $\{-1, 0, 1\}$.

We call Rt_5 the set of all ratings 5-tuples.

Define $d : Rt_5 \times Rt_5 \rightarrow \mathbb{N}$ by

$$d(\; (\; (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5) \;) \;) = \sum_{i=1}^{5} |x_i - y_i|$$

For a cluster of ratings $C = \{r_1, r_2, \cdots, r_n\} \subseteq U$, the **diameter** of the cluster is defined by:

$$diameter(C) = \max_{1 \leq i, j \leq n} (d(\; (r_i, r_j) \;))$$

(a) Calculate $diameter(\{(-1, -1, -1, -1, 1), (0, 0, 0, 0, 0), (1, 1, 1, 1, 1)\})$

(b) What's the greatest (possible) diameter of a collection that has exactly two (distinct) ratings 5-tuples?

(c) What's the least (possible) diameter of a collection that has exactly two (distinct) ratings 5-tuples?

Friday December 3

Convert $(2A)_{16}$ to

- binary (base ____)

- decimal (base ____)

- octal (base ____)

- ternary (base ____)

The bases of RNA strands are elements of the set $B = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$. The set of RNA strands S is defined (recursively) by:

$$\begin{aligned}
&\text{Basis Step:} &&\texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S\\
&\text{Recursive Step:} &&\text{If } s \in S \text{ and } b \in B, \text{ then } sb \in S
\end{aligned}$$

where sb is string concatenation.

Each of the sets below is described using set builder notation. Rewrite them using the roster method.

- $\{s \in S \mid$ the leftmost base in s is the same as the rightmost base in s and s has length 3$\}$

- $\{s \in S \mid$ there are twice as many $\texttt{A}$s as $\texttt{C}$s in s and s has length 1$\}$

Certain sequences of bases serve important biological functions in translating RNA to proteins. The following recursive definition gives a special set of RNA strands: The set of RNA strands $\hat{S}$ is defined (recursively) by

$$\begin{aligned}
&\text{Basis step:} &&\texttt{AUG} \in \hat{S}\\
&\text{Recursive step:} &&\text{If } s \in \hat{S} \text{ and } x \in R, \text{ then } sx \in \hat{S}
\end{aligned}$$

where $R = \{\texttt{UUU}, \texttt{CUC}, \texttt{AUC}, \texttt{AUG}, \texttt{GUU}, \texttt{CCU}, \texttt{GCU}, \texttt{UGG}, \texttt{GGA}\}$.

Each of the sets below is described using set builder notation. Rewrite them using the roster method.

- $\{s \in \hat{S} \mid s$ has length less than or equal to 5$\}$

- $\{s \in S \mid$ there are twice as many $\texttt{C}$s as $\texttt{A}$s in s and s has length 6$\}$

Let $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$. Consider the statement

$$\forall A \in W \ \forall B \in W \ \forall C \in W \ ((A \cap B = A \cap C) \rightarrow (B = C))$$

Translate the statement to English. Negate the statement and translate this negation to English. Decide whether the original statement or its negation is true and justify your decision.

The set of linked lists of natural numbers L is defined by

$$\text{Basis step:} \qquad\qquad\qquad\qquad [\,] \in L$$
$$\text{Recursive step:} \qquad \text{If } l \in L \text{ and } n \in \mathbb{N}, \text{ then } (n, l) \in L$$

The function $length : L \to \mathbb{N}$ that computes the length of a list is

$$\text{Basis step:} \qquad\qquad\qquad\qquad length([\,]) = 0$$
$$\text{Recursive step:} \qquad \text{If } l \in L \text{ and } n \in \mathbb{N}, \text{ then } length((n, l)) = 1 + length(l)$$

Prove or disprove: the function $length$ is onto.

Prove or disprove: the function $length$ is one-to-one.

Suppose A and B are sets and $A \subseteq B$:

True or False? If A is infinite then B is finite.

True or False? If A is countable then B is countable.

True or False? If B is infinite then A is finite.

True or False? If B is uncountable then A is countable.

Compute the last digit of

$$(42)^{2021}$$

Extra Describe the pattern that helps you perform this computation and prove it using mathematical induction.

Review

1.

   Please complete the CAPE and TA evaluations. Once you have done so complete the custom feedback form for this quarter: `https://forms.gle/pbYWSRDP2znkciM46`

   Then, (we're using the honor system here), write out the statement "I have completed the end of quarter evaluations" and you'll receive credit for this question.

Monday October 4

Find and fix any and all mistakes with the following:

(a) $(1)_2 = (1)_8$

(b) $(142)_{10} = (142)_{16}$

(c) $(20)_{10} = (10100)_2$

(d) $(35)_8 = (1D)_{16}$

Recall the definition of base expansion we discussed:

Definition For b an integer greater than 1 and n a positive integer, the **base b expansion of** n is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are nonnegative integers less than b, $a_{k-1} \neq 0$, and

$$n = \sum_{i=0}^{k-1} a_i b^i$$

Notice: *The base b expansion of a positive integer n is a string over the alphabet $\{x \in \mathbb{N} \mid x < b\}$ whose leftmost character is nonzero.*

Base b	Collection of possible coefficients in base b expansion of a positive integer
Binary ($b = 2$)	$\{0, 1\}$
Ternary ($b = 3$)	$\{0, 1, 2\}$
Octal ($b = 8$)	$\{0, 1, 2, 3, 4, 5, 6, 7\}$
Decimal ($b = 10$)	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
Hexadecimal ($b = 16$)	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ letter coefficient symbols represent numerical values $(A)_{16} = (10)_{10}$ $(B)_{16} = (11)_{10}$ $(C)_{16} = (12)_{10}$ $(D)_{16} = (13)_{10}$ $(E)_{16} = (14)_{10}$ $(F)_{16} = (15)_{10}$

We write an algorithm for converting from base b_1 expansion to base b_2 expansion:

Definition For b an integer greater than 1, w a positive integer, and n a nonnegative integer _____,
the **base b fixed-width w expansion of n** is

$$(a_{w-1} \cdots a_1 a_0)_{b,w}$$

where $a_0, a_1, \ldots, a_{w-1}$ are nonnegative integers less than b and

$$n = \sum_{i=0}^{w-1} a_i b^i$$

Decimal $b = 10$	Binary $b = 2$	Binary fixed-width 10 $b = 2,\ w = 10$	Binary fixed-width 7 $b = 2,\ w = 7$	Binary fixed-width 4 $b = 2,\ w = 4$
$(20)_{10}$				
	(a)	(b)	(c)	(d)

Definition For b an integer greater than 1, w a positive integer, w' a positive integer, and x a real number
the **base b fixed-width expansion of x with integer part width w and fractional part width w'**
is $(a_{w-1} \cdots a_1 a_0 . c_1 \cdots c_{w'})_{b,w,w'}$ where $a_0, a_1, \ldots, a_{w-1}, c_1, \ldots, c_{w'}$ are nonnegative integers less than b and

$$x \geq \sum_{i=0}^{w-1} a_i b^i + \sum_{j=1}^{w'} c_j b^{-j} \qquad \text{and} \qquad x < \sum_{i=0}^{w-1} a_i b^i + \sum_{j=1}^{w'} c_j b^{-j} + b^{-w'}$$

3.75 in fixed-width binary, integer part width 2, fractional part width 8	
0.1 in fixed-width binary, integer part width 2, fractional part width 8	

```
welcome $jshell
|   Welcome to JShell -- Version 10.0.1
|   For an introduction type: /help intro

jshell> 0.1
$1 ==>

jshell> 0.2
$2 ==>

jshell> 0.1 + 0.2
$3 ==>

jshell> Math.sqrt(2)
$4 ==>

jshell> Math.sqrt(2)*Math.sqrt(2)
$5 ==>

jshell>
```

Note: Java uses floating point, not fixed width representation, but similar rounding errors appear in both.

Review: Week 2 Monday

1. Recall the definitions from class for number representations for **base b expansion of n**, **base b fixed-width w expansion of n**, and **base b fixed-width expansion of x with integer part width w and fractional part width w'**.

   For example, the base 2 (binary) expansion of 4 is $(100)_2$ and the base 2 (binary) fixed-width 8 expansion of 4 is $(00000100)_{2,8}$ and the base 2 (binary) fixed-width expansion of 4 with integer part width 3 and fractional part width 2 of 4 is $(100.00)_{2,3,2}$

   Compute the listed expansions. Enter your number using the notation for base expansions with parentheses but without subscripts. For example, if your answer were $(100)_{2,3}$ you would type `(100)2,3` into Gradescope.

   (a) Give the binary (base 2) expansion of the number whose octal (base 8) expansion is

   $$(371)_8$$

   (b) Give the decimal (base 10) expansion of the number whose octal (base 8) expansion is

   $$(371)_8$$

   (c) Give the octal (base 8) fixed-width 3 expansion of $(9)_{10}$?

   (d) Give the ternary (base 3) fixed-width 8 expansion of $(9)_{10}$?

   (e) Give the hexadecimal (base 16) fixed-width 6 expansion of $(16711935)_{10}$?[3]

   (f) Give the hexadecimal (base 16) fixed-width 4 expansion of

   $$(1011\ 1010\ 1001\ 0000)_2$$

   Note: the spaces between each group of 4 bits above are for your convenience only. How might they help your calculations?

   (g) Give the binary fixed width expansion of 0.125 with integer part width 2 and fractional part width 4.

   (h) Give the binary fixed width expansion of 1 with integer part width 2 and fractional part width 3.

2. Select all and only the correct choices below.

   (a) Suppose you were told that the positive integer n_1 has the property that n_1 **div** $2 = 0$. Which of the following can you conclude?

      i. n_1 has a binary (base 2) expansion
      ii. n_1 has a ternary (base 3) expansion
      iii. n_1 has a hexadecimal (base 16) expansion
      iv. n_1 has a base 2 fixed-width 1 expansion
      v. n_1 has a base 2 fixed-width 20 expansion

   (b) Suppose you were told that the positive integer n_2 has the property that n_2 **mod** $4 = 0$. Which of the following can you conclude?

---

[3]This matches a frequent debugging task – sometimes a program will show a number formatted as a base 10 integer that is much better understood with another representation.

i. the leftmost symbol in the binary (base 2) expansion of n_2 is 1

ii. the leftmost symbol in the base 4 expansion of n_2 is 1

iii. the rightmost symbol in the base 4 expansion of n_2 is 0

iv. the rightmost symbol in the octal (base 8) expansion of n_2 is 0

Wednesday October 6

base b expansion of n	base b fixed-width w expansion of n
For b an integer greater than 1 and n a positive integer, the **base b expansion of** n is $(a_{k-1}\cdots a_1 a_0)_b$ where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are nonnegative integers less than b, $a_{k-1} \neq 0$, and $n = a_{k-1}b^{k-1} + \cdots + a_1 b + a_0$	For b an integer greater than 1, w a positive integer, and n a nonnegative integer with $n < b^w$, the **base b fixed-width w expansion of** n is $(a_{w-1}\cdots a_1 a_0)_{b,w}$ where $a_0, a_1, \ldots, a_{w-1}$ are nonnegative integers less than b and $n = a_{w-1}b^{w-1} + \cdots + a_1 b + a_0$

Representing negative integers in binary: Fix a positive integer width for the representation w, $w > 1$.

	To represent a positive integer n	To represent a negative integer $-n$
Sign-magnitude	$[0a_{w-2}\cdots a_0]_{s,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $n = 17$, $w = 7$:	$[1a_{w-2}\cdots a_0]_{s,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $-n = -17$, $w = 7$:
2s complement	$[0a_{w-2}\cdots a_0]_{2c,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $n = 17$, $w = 7$:	$[1a_{w-2}\cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $-n = -17$, $w = 7$:
Extra example: 1s complement	$[0a_{w-2}\cdots a_0]_{1c,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $n = 17$, $w = 7$:	$[1\bar{a}_{w-2}\cdots \bar{a}_0]_{1c,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ and we define $\bar{0} = 1$ and $\bar{1} = 0$. Example $-n = -17$, $w = 7$:

For positive integer n, to represent $-n$ in 2s complement with width w,

- Calculate $2^{w-1} - n$, convert result to binary fixed-width $w - 1$, pad with leading 1, or

- Express $-n$ as a sum of powers of 2, where the leftmost 2^{w-1} is negative weight, or

- Convert n to binary fixed-width w, flip bits, add 1 (ignore overflow)

Challenge: use definitions to explain why each of these approaches works.

Representing 0:

So far, we have representations for positive and negative integers. What about 0?

	To represent a **non-negative** integer n	To represent a **non-positive** integer $-n$
Sign-magnitude	$[0a_{w-2}\cdots a_0]_{s,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $n = 0$, $w = 7$: (a)	$[1a_{w-2}\cdots a_0]_{s,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $-n = 0$, $w = 7$: (b)
2s complement	$[0a_{w-2}\cdots a_0]_{2c,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $n = 0$, $w = 7$: (c)	$[1a_{w-2}\cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $-n = 0$, $w = 7$: (d)

Fixed-width addition: adding one bit at time, using the usual column-by-column and carry arithmetic, and dropping the carry from the leftmost column so the result is the same width as the summands. *Does this give the right value for the sum?*

$$(1\ 1\ 0\ 1\ 0\ 0)_{2,6}$$
$$+(0\ 0\ 0\ 1\ 0\ 1)_{2,6}$$

$$[1\ 1\ 0\ 1\ 0\ 0]_{s,6}$$
$$+[0\ 0\ 0\ 1\ 0\ 1]_{s,6}$$

$$[1\ 1\ 0\ 1\ 0\ 0]_{2c,6}$$
$$+[0\ 0\ 0\ 1\ 0\ 1]_{2c,6}$$

Review: Week 2 Wednesday

1. Recall the definitions of signed integer representations from class: sign-magnitude and 2s complement.

   (a) Give the 2s complement width 6 representation of the number represented in binary fixed-width 5 representation as $(00101)_{2,5}$.

   (b) Give the 2s complement width 6 representation of the number represented in binary fixed-width 5 representation as $(10101)_{2,5}$.

   (c) Give the 2s complement width 4 representation of the number represented in sign-magnitude width 4 as $[1111]_{s,4}$.

   (d) Give the sign magnitude width 4 representation of the number represented in 2s complement width 4 as $[1111]_{2c,4}$.

   (e) Give the sign magnitude width 6 representation of the number represented in sign magnitude width 4 as $[1111]_{s,4}$.

   (f) Give the 2s complement width 6 representation of the number represented in 2s complement width 4 as $[1111]_{2c,4}$.

2. Recall the definitions of signed integer representations from class: sign-magnitude and 2s complement.

   (a) In binary fixed-width addition (adding one bit at time, using the usual column-by-column and carry arithmetic, and ignoring the carry from the leftmost column), we get:

$$
\begin{array}{ll}
1110 & \text{first summand} \\
+0100 & \text{second summand} \\
\hline
0010 & \text{result}
\end{array}
$$

   Select all and only the true statements below:
   - i. When interpreting each of the summands and the result in binary fixed-width 4, the result represents the actual value of the sum of the summands.
   - ii. When interpreting each of the summands and the sum in sign-magnitude width 4, the result represents the actual value of the sum of the summands.
   - iii. When interpreting each of the summands and the sum in 2s complement width 4, the result represents the actual value of the sum of the summands.

   (b) In binary fixed-width addition (adding one bit at time, using the usual column-by-column and carry arithmetic, and ignoring the carry from the leftmost column), we get:

$$
\begin{array}{ll}
0110 & \text{first summand} \\
+0111 & \text{second summand} \\
\hline
1101 & \text{result}
\end{array}
$$

   Select all and only the true statements below:
   - i. When interpreting each of the summands and the result in binary fixed-width 4, the result represents the actual value of the sum of the summands.
   - ii. When interpreting each of the summands and the sum in sign-magnitude width 4, the result represents the actual value of the sum of the summands.
   - iii. When interpreting each of the summands and the sum in 2s complement width 4, the result represents the actual value of the sum of the summands.

Friday October 8

In a **combinatorial circuit** (also known as a **logic circuit**), we have **logic gates** connected by **wires**. The inputs to the circuits are the values set on the input wires: possible values are 0 (low) or 1 (high). The values flow along the wires from left to right. A wire may be split into two or more wires, indicated with a filled-in circle (representing solder). Values stay the same along a wire. When one or more wires flow into a gate, the output value of that gate is computed from the input values based on the gate's definition table. Outputs of gates may become inputs to other gates.
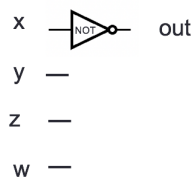
| Inputs | | Output |
x	y	x AND y
1	1	1
1	0	0
0	1	0
0	0	0
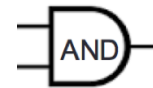
| Inputs | | Output |
x	y	x XOR y
1	1	0
1	0	1
0	1	1
0	0	0

| Input | Output |
x	NOT x
1	0
0	1

Example digital circuit:

Output when $x = 1, y = 0, z = 0, w = 1$ is _____
Output when $x = 1, y = 1, z = 1, w = 1$ is _____
Output when $x = 0, y = 0, z = 0, w = 1$ is _____

Draw a logic circuit with inputs x and y whose output is always 0. *Can you use exactly 1 gate?*

Fixed-width addition: adding one bit at time, using the usual column-by-column and carry arithmetic, and dropping the carry from the leftmost column so the result is the same width as the summands. In many cases, this gives representation of the correct value for the sum when we interpret the summands in fixed-width binary or in 2s complement.

For single column:

Input		Output	
x_0	y_0	c_0	s_0
1	1		
1	0		
0	1		
0	0		
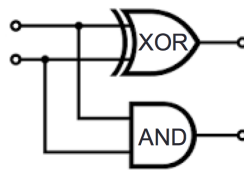
Draw a logic circuit that implements binary addition of two numbers that are each represented in fixed-width binary:

- Inputs x_0, y_0, x_1, y_1 represent $(x_1x_0)_{2,2}$ and $(y_1y_0)_{2,2}$

- Outputs z_0, z_1, z_2 represent $(z_2z_1z_0)_{2,3} = (x_1x_0)_{2,2} + (y_1y_0)_{2,2}$ (may require up to width 3)
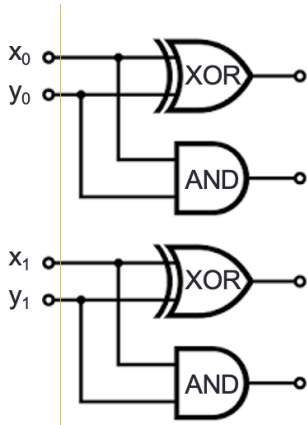
First approach: half-adder for each column, then combine carry from right column with sum of left column

Write expressions for the circuit output values in terms of input values:

$z_0 = $ _____

$z_1 = $ _____

$z_2 = $ _____

Second approach: for middle column, first add carry from right column to x_1, then add result to y_1

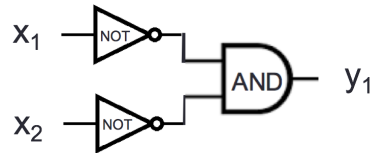Write expressions for the circuit output values in terms of input values:

$z_0 = $ _____

$z_1 = $ _____

$z_2 = $ _____

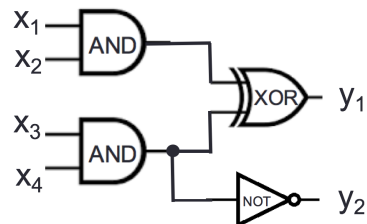*Extra example* Describe how to generalize this addition circuit for larger width inputs.

Review: Week 2 Friday

1. (a) Consider the logic circuit

Calculate the value of the output of this circuit (y_1) for each of the following settings(s) of input values.

i. $x_1 = 1$, $x_2 = 1$

ii. $x_1 = 1$, $x_2 = 0$

iii. $x_1 = 0$, $x_2 = 1$

iv. $x_1 = 0$, $x_2 = 0$

(b) Consider the logic circuit

For which of the following settings(s) of input values is the output $y_1 = 0$, $y_2 = 1$? (Select all and only those that apply.)

i. $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, and $x_4 = 0$

ii. $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, and $x_4 = 1$

iii. $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, and $x_4 = 1$

iv. $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, and $x_4 = 1$

2. Recall this circuit from class:
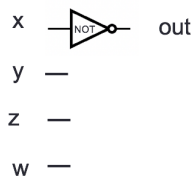
Which of the following is true about all possible input values x, y, z, w? (Select all and only choices that are true for all values.)

(a) The output *out* is set to 1 exactly when x is 0, and it is set to 0 otherwise.

(b) The output *out* is set to 1 exactly when $(xyzw)_{2,4} < 8$, and it is set to 0 otherwise.

(c) The output *out* is set to 1 exactly when $(wzyx)_{2,4}$ is an even integer, and it is set to 0 otherwise.

Monday October 11

Logical operators aka propositional connectives

Conjunction	AND	$\land$	`\land`	2 inputs	Evaluates to T exactly when **both** inputs are T
Exclusive or	XOR	$\oplus$	`\oplus`	2 inputs	Evaluates to T exactly when **exactly one** of inputs is T
Disjunction	OR	$\lor$	`\lor`	2 inputs	Evaluates to T exactly when **at least one** of inputs is T
Negation	NOT	$\lnot$	`\lnot`	1 input	Evaluates to T exactly when its input is F

Truth tables: Input-output tables where we use T for 1 and F for 0.

Input		Output		
		Conjunction	**Exclusive or**	**Disjunction**
p	q	$p \land q$	$p \oplus q$	$p \lor q$
T	T	T	F	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F

Input	Output
	Negation
p	$\lnot p$
T	F
F	T

Input			Output		
p	q	r		$(p \land q) \oplus (\ (p \oplus q) \land r\ )$	$(p \land q) \lor (\ (p \oplus q) \land r\ )$
T	T	T			
T	T	F			
T	F	T			
T	F	F			
F	T	T			
F	T	F			
F	F	T			
F	F	F			

Given a truth table, how do we find an expression using the input variables and logical operators that has the output values specified in this table?

Application: design a circuit given a desired input-output relationship.

| Input | | Output | |
p	q	$mystery_1$	$mystery_2$
T	T	T	F
T	F	T	F
F	T	F	F
F	F	T	T

Expressions that have output $mystery_1$ are

Expressions that have output $mystery_2$ are

Definition An expression built of variables and logical operators is in **disjunctive normal form** (DNF) means that it is an OR of ANDs of variables and their negations.

Definition An expression built of variables and logical operators is in **conjunctive normal form** (CNF) means that it is an AND of ORs of variables and their negations.

Extra example: An expression that has output ? is:

| Input | | | Output |
p	q	r	?
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	F

Review: Week 3 Monday

1. (a) Consider the logic circuit
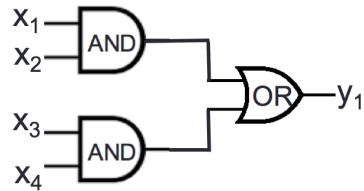
For which of the following settings(s) of input values is the output $y_1 = 0$? (Select all and only those that apply.)

   i. $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, and $x_4 = 0$
   ii. $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, and $x_4 = 1$
   iii. $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, and $x_4 = 1$
   iv. $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, and $x_4 = 1$

   (b) Consider the logic circuits
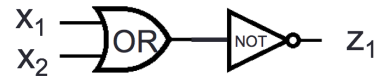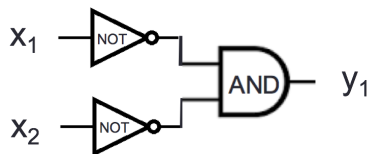
For which of the following settings(s) of input values do the outputs of these circuits have the same value, i.e. $y_1 = z_1$? (Select all and only those that apply.)

   i. $x_1 = 1$, $x_2 = 1$
   ii. $x_1 = 1$, $x_2 = 0$
   iii. $x_1 = 0$, $x_2 = 1$
   iv. $x_1 = 0$, $x_2 = 0$

2. For each of the following compound propositions, determine if it is in DNF, CNF, both, or neither.

   (a) $(x \lor y \lor z) \land (x \land \neg y \land z)$
   (b) $\neg(x \land y \land z) \land \neg(\neg x \land y \land \neg z)$

Wednesday October 13

Proposition: Declarative sentence that is true or false (not both).

Propositional variable: Variable that represents a proposition.

Compound proposition: New proposition formed from existing propositions (potentially) using logical operators. *Note*: A propositional variable is one example of a compound proposition.

Truth table: Table with one row for each of the possible combinations of truth values of the input and an additional column that shows the truth value of the result of the operation corresponding to a particular row.

Logical equivalence : Two compound propositions are **logically equivalent** means that they have the same truth values for all settings of truth values to their propositional variables.

Tautology: A compound proposition that evaluates to true for all settings of truth values to its propositional variables; it is abbreviated T.

Contradiction: A compound proposition that evaluates to false for all settings of truth values to its propositional variables; it is abbreviated F.

Contingency: A compound proposition that is neither a tautology nor a contradiction.

Label each of the following as a tautology, contradiction, or contingency.

$p \wedge p$

$p \oplus p$

$p \vee p$

$p \vee \neg p$

$p \wedge \neg p$

Extra Example: Which of the compound propositions in the table below are logically equivalent?

Input		Output				
p	q	$\neg(p \wedge \neg q)$	$\neg(\neg p \vee \neg q)$	$(\neg p \vee q)$	$(\neg q \vee \neg p)$	$(p \wedge q)$
T	T					
T	F					
F	T					
F	F					

Input		Output				
		Conjunction	Exclusive or	Disjunction	Conditional	Biconditional
p	q	$p \wedge q$	$p \oplus q$	$p \vee q$	$p \to q$	$p \leftrightarrow q$
T	T	T	F	T	T	T
T	F	F	T	T	F	F
F	T	F	T	T	T	F
F	F	F	F	F	T	T
		"p and q"	"p xor q"	"p or q"	"if p then q"	"p if and only if q"

The only way to make the conditional statement $p \to q$ false is to _____

The **hypothesis** of $p \to q$ is _____   The **antecedent** of $p \to q$ is _____

The **conclusion** of $p \to q$ is _____   The **consequent** of $p \to q$ is _____

The **converse** of $p \to q$ is _____

The **inverse** of $p \to q$ is _____

The **contrapositive** of $p \to q$ is _____

We can use a recursive definition to describe all compound propositions that use propositional variables from a specified collection. Here's the definition for all compound propositions whose propositional variables are in $\{p, q\}$.

Basis Step:	p and q are each a compound proposition
Recursive Step:	If x is a compound proposition then so is $(\neg x)$ and if x and y are both compound propositions then so is each of $(x \wedge y), (x \oplus y), (x \vee y), (x \to y), (x \leftrightarrow y)$

Order of operations (Precedence) for logical operators:

Negation, then conjunction / disjunction, then conditional / biconditionals.

Example: $\neg p \vee \neg q$ means $(\neg p) \vee (\neg q)$.

(Some) logical equivalences

Can replace p and q with any compound proposition

$\neg(\neg p) \equiv p$ **Double negation**

$p \lor q \equiv q \lor p$ $p \land q \equiv q \land p$ **Commutativity** Ordering of terms

$(p \lor q) \lor r \equiv p \lor (q \lor r)$ $(p \land q) \land r \equiv p \land (q \land r)$ **Associativity** Grouping of terms

$p \land F \equiv F$ $p \lor T \equiv T$ $p \land T \equiv p$ $p \lor F \equiv p$ **Domination** aka short circuit evaluation

$\neg(p \land q) \equiv \neg p \lor \neg q$ $\neg(p \lor q) \equiv \neg p \land \neg q$ **DeMorgan's Laws**

$p \to q \equiv \neg p \lor q$

$p \to q \equiv \neg q \to \neg p$ **Contrapositive**

$\neg(p \to q) \equiv p \land \neg q$

$\neg(p \leftrightarrow q) \equiv p \oplus q$

$p \leftrightarrow q \equiv q \leftrightarrow p$

Extra examples:

$p \leftrightarrow q$ is not logically equivalent to $p \land q$ because _____

$p \to q$ is not logically equivalent to $q \to p$ because _____

Review: Week 3 Wednesday

1. For each of the following propositions, indicate exactly one of:

   - There is no assignment of truth values to its variables that makes it true,
   - There is exactly one assignment of truth values to its variables that makes it true, or
   - There are exactly two assignments of truth values to its variables that make it true, or
   - There are exactly three assignments of truth values to its variables that make it true, or
   - *All* assignments of truth values to its variables make it true.

   (a) $x \wedge y \wedge (x \vee y)$

   (b) $\neg x \wedge y \wedge (x \vee y)$

   (c) $x \wedge \neg y \wedge (x \wedge y)$

   (d) $\neg x \wedge (y \vee \neg y)$

   (e) $x \wedge (y \vee \neg x)$

   For each of the following propositions, indicate exactly one of:

2.
   - There is no assignment of truth values to its variables that makes it true,
   - There is exactly one assignment of truth values to its variables that makes it true, or
   - There are exactly two assignments of truth values to its variables that make it true, or
   - There are exactly three assignments of truth values to its variables that make it true, or
   - *All* assignments of truth values to its variables make it true.

   (a) $(p \leftrightarrow q) \oplus (p \wedge q)$

   (b) $(p \rightarrow q) \vee (q \rightarrow p)$

   (c) $(p \rightarrow q) \wedge (q \rightarrow p)$

   (d) $\neg(p \rightarrow q)$

Friday October 15

Common ways to express logical operators in English:

Negation $\neg p$ can be said in English as
- Not p.

- It's not the case that p.

- p is false.

Conjunction $p \wedge q$ can be said in English as
- p and q.

- Both p and q are true.

- p but q.

Exclusive or $p \oplus q$ can be said in English as
- p or q, but not both.

- Exactly one of p and q is true.

Disjunction $p \vee q$ can be said in English as
- p or q, or both.

- p or q (inclusive).

- At least one of p and q is true.

Conditional $p \to q$ can be said in English as

- if p, then q.

- p is sufficient for q.

- q when p.

- q whenever p.

- p implies q.

- q follows from p.

- p is sufficient for q.

- q is necessary for p.

- p only if q.

Biconditional
- p if and only if q.

- p iff q.

- If p then q, and conversely.

- p is necessary and sufficient for q.

Translation: Express each of the following sentences as compound propositions, using the given propositions.

"A sufficient condition for the warranty to be good is that you bought the computer less than a year ago"

w is "the warranty is good"

b is "you bought the computer less than a year ago"

"Whenever the message was sent from an unknown system, it is scanned for viruses."

s is "The message is scanned for viruses"

u is "The message was sent from an unknown system"

"I will complete my to-do list only if I put a reminder in my calendar"

d is "I will complete my to-do list"

c is "I put a reminder in my calendar"

Definition: A collection of compound propositions is called **consistent** if there is an assignment of truth values to the propositional variables that makes each of the compound propositions true.

Consistency:

Whenever the system software is being upgraded, users cannot access the file system. If users can access the file system, then they can save new files. If users cannot save new files, then the system software is not being upgraded.

1. Translate to symbolic compound propositions

2. Look for some truth assignment to the propositional variables for which all the compound propositions output T

Review: Week 3 Friday

1. Express each of the following sentences as compound propositions, using the given propositions.

   (a) "If you try to run Zoom while your computer is running many applications, the video is likely to be choppy and laggy." t is "you run Zoom while your computer is running many applications", c is "the video is likely to be choppy", g is "the video is likely to be laggy"

      i. $t \rightarrow (c \wedge g)$
      ii. $(c \wedge g) \rightarrow t$
      iii. $(c \wedge g) \leftrightarrow t$
      iv. $t \oplus (c \wedge g)$

   (b) "To connect wirelessly on campus without logging in you need to use the UCSD-Guest network." c is "connect wirelessly on campus", g is "logging in", and u is "use UCSD-Guest network".

      i. $c \wedge \neg g \wedge u$
      ii. $(c \wedge \neg g) \vee u$
      iii. $(c \wedge \neg g) \oplus u$
      iv. $(c \wedge \neg g) \rightarrow u$
      v. $u \rightarrow (c \wedge \neg g)$
      vi. $u \leftrightarrow (c \wedge \neg g)$

   For each of the following system specifications, identify the compound propositions that give their translations to logic and then determine if the translated collection of compound propositions is consistent.

2. (a) Specification: If the computer is out of memory, then network connectivity is unreliable. No disk errors can occur when the computer is out of memory. Disk errors only occur when network connectivity is unreliable.

   Translation: $M = $ "the computer is out of memory"; $N = $ "network connectivity is unreliable"; $D = $ "disk errors can occur".

      i.
      $$\neg M \rightarrow N$$
      $$\neg D \rightarrow M$$
      $$D \rightarrow N$$

      ii.
      $$M \rightarrow \neg N$$
      $$\neg D \wedge M$$
      $$N \rightarrow D$$

      iii.
      $$M \rightarrow N$$
      $$M \rightarrow \neg D$$
      $$\neg N \rightarrow \neg D$$

(b) Specification: Whether you think you can, or you think you can't - you're right. [4]

Translation: $T = $ "you think you can"; $C = $ "you can".

i.

$$T \rightarrow C$$
$$\neg T \rightarrow \neg C$$

ii.

$$T \wedge C$$
$$\neg T \wedge \neg C$$

iii.

$$T \rightarrow \neg T$$
$$C \rightarrow \neg C$$

(c) Specification: A secure password must be private and complicated. If a password is complicated then it will be hard to remember. People write down hard-to-remember passwords. If a password is written down, it's not private. The password is secure.

Translation: $S = $ "the password is secure"; $P = $ "the password is private"; $C = $ "the password is complicated"; $H = $ "the password is hard to remember"; $W = $ "the password is written down".

i.

$$\neg(P \wedge C) \rightarrow \neg S$$
$$C \rightarrow H$$
$$W \wedge H$$
$$W \rightarrow \neg P$$
$$S$$

ii.

$$(P \wedge C) \rightarrow S$$
$$C \rightarrow H$$
$$W \rightarrow H$$
$$W \rightarrow P$$
$$S$$

iii.

$$S \rightarrow (P \wedge C)$$
$$C \rightarrow H$$
$$H \rightarrow W$$
$$W \rightarrow \neg P$$
$$S$$

---

[4]Henry Ford

Monday October 18

Real-life representations are often prone to corruption. Biological codes, like RNA, may mutate naturally[5] and during measurement; cosmic radiation and other ambient noise can flip bits in computer storage[6]. One way to recover from corrupted data is to introduce or exploit redundancy.

Consider the following algorithm to introduce redundancy in a string of 0s and 1s.

Create redundancy by repeating each bit three times

```
1  procedure redun3(a_{k-1} ··· a_0: a nonempty bitstring)
2  for i := 0 to k-1
3      c_{3i}   := a_i
4      c_{3i+1} := a_i
5      c_{3i+2} := a_i
6  return c_{3k-1} ··· c_0
```

Decode sequence of bits using majority rule on consecutive three bit sequences

```
1  procedure decode3(c_{3k-1} ··· c_0: a nonempty bitstring whose length is an integer multiple of 3)
2  for i := 0 to k-1
3      if exactly two or three of c_{3i}, c_{3i+1}, c_{3i+2} are set to 1
4          a_i := 1
5      else
6          a_i := 0
7  return a_{k-1} ··· a_0
```

Give a recursive definition of the set of outputs of the $redun3$ procedure, Out,

Consider the message $m = 0001$ so that the sender calculates $redun3(m) = redun3(0001) = 000000000111$.

Introduce ____ errors into the message so that the signal received by the receiver is _____ but the receiver is still able to decode the original message.

Challenge: what is the biggest number of errors you can introduce?

Building a circuit for lines 3-6 in *decode* procedure: given three input bits, we need to determine whether the majority is a 0 or a 1.

c_{3i}	c_{3i+1}	c_{3i+2}	a_i
1	1	1	
1	1	0	
1	0	1	
1	0	0	
0	1	1	
0	1	0	
0	0	1	
0	0	0	

Circuit

---

[5]Mutations of specific RNA codons have been linked to many disorders and cancers.

[6]This RadioLab podcast episode goes into more detail on bit flips: `https://www.wnycstudios.org/story/bit-flip`

Definition: The **Cartesian product** of the sets A and B, $A \times B$, is the set of all ordered pairs (a, b), where $a \in A$ and $b \in B$. That is: $A \times B = \{(a, b) \mid (a \in A) \wedge (b \in B)\}$. The Cartesian product of the sets $A_1, A_2, \ldots, A_n$, denoted by $A_1 \times A_2 \times \cdots \times A_n$, is the set of ordered n-tuples $(a_1, a_2, \ldots, a_n)$, where a_i belongs to A_i for $i = 1, 2, \ldots, n$. That is,

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i \text{ for } i = 1, 2, \ldots, n\}$$

Recall that S is defined as the set of all RNA strands, nonempty strings made of the bases in $B = \{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}\}$. We define the functions

$$mutation : S \times \mathbb{Z}^+ \times B \to S \qquad insertion : S \times \mathbb{Z}^+ \times B \to S$$

$$deletion : \{s \in S \mid rnalen(s) > 1\} \times \mathbb{Z}^+ \to S \qquad \text{with rules}$$

```
1  procedure mutation(b₁···bₙ: a RNA strand, k: a positive integer, b: an element of B)
2  for i := 1 to n
3     if i = k
4        cᵢ := b
5     else
6        cᵢ := bᵢ
7  return c₁···cₙ  {The return value is a RNA strand made of the cᵢ values}
```

```
1   procedure insertion(b₁···bₙ: a RNA strand, k: a positive integer, b: an element of B)
2   if k > n
3      for i := 1 to n
4         cᵢ := bᵢ
5      cₙ₊₁ := b
6   else
7      for i := 1 to k − 1
8         cᵢ := bᵢ
9      cₖ := b
10     for i := k + 1 to n + 1
11        cᵢ := bᵢ₋₁
12  return c₁···cₙ₊₁  {The return value is a RNA strand made of the cᵢ values}
```

```
1   procedure deletion(b₁···bₙ: a RNA strand with n > 1, k: a positive integer)
2   if k > n
3      m := n
4      for i := 1 to n
5         cᵢ := bᵢ
6   else
7      m := n − 1
8      for i := 1 to k − 1
9         cᵢ := bᵢ
10     for i := k to n − 1
11        cᵢ := bᵢ₊₁
12  return c₁···cₘ  {The return value is a RNA strand made of the cᵢ values}
```

Trace the pseudocode to find the output of $mutation(\,(\texttt{AUC}, 3, \texttt{G})\,)$

Fill in the blanks so that $insertion(\,(\texttt{AUC}, \_, \_)\,) = \texttt{AUCG}$

Fill in the blanks so that $deletion(\,(\_, \_)\,) = \texttt{G}$

Review

1.

   In this question, we will consider how to build a logic circuit with inputs x_0, y_0, x_1, y_1 and output z such that $z = 1$ exactly when $(x_1 x_0)_{2,2} < (y_1 y_0)_{2,2}$ and $z = 0$ exactly when $(x_1 x_0)_{2,2} \geq (y_1 y_0)_{2,2}$.

   (a) The first step towards designing this logic circuit is to construct its input-output table. How many rows does this table have (not including the header row labelling the columns)?

   (b) What is the output for the row whose input values are $x_0 = 0$, $y_0 = 1$, $x_1 = 1$, $y_1 = 0$?

   (c) What is the output for the row whose input values are $x_0 = 0$, $y_0 = 1$, $x_1 = 0$, $y_1 = 1$?

2.

   Recall the procedures $redun3$ and $decode3$ from class.

   (a) Give the output of $redun3(100)$.

   (b) If the output of running $redun3$ is 000000111000111, what was its input?

   (c) Give the output of $decode3(100)$.

   (d) How many distinct possible inputs to $decode3$ give the output 01?

3.

   Recall the procedures $mutation$ and $insertion$ and $deletion$ from class.

   (a) Trace the pseudocode to find the output of $mutation(\ (\mathtt{AUC}, 2, \mathtt{U})\ )$

   (b) Trace the pseudocode to find the output of $insertion(\ (\mathtt{AUC}, 1, \mathtt{G})\ )$

   (c) Trace the pseudocode to find the output of $deletion(\ (\mathtt{AUC}, 1)\ )$

Wednesday October 20

Definition: A **predicate** is a function from a given set (domain) to $\{T, F\}$.

A predicate can be applied, or **evaluated** at, an element of the domain.

Usually, a predicate *describes a property* that domain elements may or may not have.

Two predicates over the same domain are **equivalent** means they evaluate to the same truth values for all possible assignments of domain elements to the input. In other words, they are equivalent means that they are equal as functions.

To define a predicate, we must specify its domain and its value at each domain element. The rule assigning truth values to domain elements can be specified using a formula, English description, in a table (if the domain is finite), or recursively (if the domain is recursively defined).

| Input | Output | | |
| | $V(x)$ | $N(x)$ | $Mystery(x)$ |
x	$[x]_{2c,3} > 0$	$[x]_{2c,3} < 0$	
000	F		T
001	T		T
010	T		T
011	T		F
100	F		F
101	F		T
110	F		F
111	F		T

The domain for each of the predicates $V(x)$, $N(x)$, $Mystery(x)$ is _____ .

Fill in the table of values for the predicate $N(x)$ based on the formula given.

Definition: The **truth set** of a predicate is the collection of all elements in its domain where the predicate evaluates to T.

Notice that specifying the domain and the truth set is sufficient for defining a predicate.

The truth set for the predicate $V(x)$ is _____ .

The truth set for the predicate $N(x)$ is _____ .

The truth set for the predicate $Mystery(x)$ is _____ .

The **universal quantification** of predicate $P(x)$ over domain U is the statement "$P(x)$ for all values of x in the domain U" and is written $\forall x P(x)$ or $\forall x \in U \; P(x)$. When the domain is finite, universal quantification over the domain is equivalent to iterated *conjunction* (ands).

The **existential quantification** of predicate $P(x)$ over domain U is the statement "There exists an element x in the domain U such that $P(x)$" and is written $\exists x P(x)$ for $\exists x \in U \; P(x)$. When the domain is finite, existential quantification over the domain is equivalent to iterated *disjunction* (ors).

An element for which $P(x) = F$ is called a **counterexample** of $\forall x P(x)$.

An element for which $P(x) = T$ is called a **witness** of $\exists x P(x)$.

Statements involving predicates and quantifiers are **logically equivalent** means they have the same truth value no matter which predicates (domains and functions) are substituted in.

Quantifier version of De Morgan's laws: $\boxed{\neg \forall x P(x) \;\equiv\; \exists x \, (\neg P(x))}$ $\qquad$ $\boxed{\neg \exists x Q(x) \;\equiv\; \forall x \, (\neg Q(x))}$

Examples of quantifications using $V(x), N(x), Mystery(x)$:

True or **False**: $\exists x \; (\; V(x) \wedge N(x) \;)$

True or **False**: $\forall x \; (\; V(x) \rightarrow N(x) \;)$

True or **False**: $\exists x \; (\; N(x) \leftrightarrow Mystery(x) \;)$

Rewrite $\neg \forall x \; (\; V(x) \oplus Mystery(x) \;)$ into a logical equivalent statement.

Notice that these are examples where the predicates have *finite* domain. How would we evaluate quantifications where the domain may be infinite?

Example predicates on S, the set of RNA strands (an infinite set)

$H : S \rightarrow \{T, F\}$ where $H(s) = T$ for all s.

Truth set of H is _____

$F_{\mathtt{A}} : S \rightarrow \{T, F\}$ defined recursively by:

  Basis step: $F_{\mathtt{A}}(\mathtt{A}) = T$, $F_{\mathtt{A}}(\mathtt{C}) = F_{\mathtt{A}}(\mathtt{G}) = F_{\mathtt{A}}(\mathtt{U}) = F$

  Recursive step: If $s \in S$ and $b \in B$, then $F_{\mathtt{A}}(sb) = F_{\mathtt{A}}(s)$.

Example where $F_{\mathtt{A}}$ evaluates to T is _____

Example where $F_{\mathtt{A}}$ evaluates to F is _____

Review

1.

Recall the predicates $V(x)$, $N(x)$, and $Mystery(x)$ on domain $\{000, 001, 010, 011, 100, 101, 110, 111\}$ from class. Which of the following is true? (Select all and only that apply.)

(a) $(\forall x\ V(x)) \vee (\forall x\ N(x))$

(b) $(\exists x\ V(x)) \wedge (\exists x\ N(x)) \wedge (\exists x\ Mystery(x))$

(c) $\exists x\ (\ V(x) \wedge N(x) \wedge Mystery(x)\ )$

(d) $\forall x\ (\ V(x) \oplus N(x)\ )$

(e) $\forall x\ (\ Mystery(x) \rightarrow V(x)\ )$

2.

Consider the following predicates, each of which has as its domain the set of all bitstrings whose leftmost bit is 1

$E(x)$ is T exactly when $(x)_2$ is even, and is F otherwise

$L(x)$ is T exactly when $(x)_2 < 3$, and is F otherwise

$M(x)$ is T exactly when $(x)_2 > 256$ and is F otherwise.

(a) What is $E(110)$?

(b) Why is $L(00)$ undefined?

    i. Because the domain of L is infinite

    ii. Because 00 does not have 1 in the leftmost position

    iii. Because 00 has length 2, not length 3

    iv. Because $(00)_{2,2} = 0$ which is less than 3

(c) Is there a bitstring of width (where width is the number of bits) 6 at which $M(x)$ evaluates to T?

3.

For this question, we will use the following predicate.

F_A with domain S is defined recursively by:

    Basis step: $F_A(A) = T$, $F_A(C) = F_A(G) = F_A(U) = F$

    Recursive step: If $s \in S$ and $b \in B$, then $F_A(sb) = F_A(s)$

Which of the following is true? (Select all and only that apply.)

(a) $F_A(\mathtt{AA})$

(b) $F_A(\mathtt{AC})$

(c) $F_A(\mathtt{AG})$

(d) $F_A(\mathtt{AU})$

(e) $F_{\mathtt{A}}(\mathtt{CA})$

(f) $F_{\mathtt{A}}(\mathtt{CC})$

(g) $F_{\mathtt{A}}(\mathtt{CG})$

(h) $F_{\mathtt{A}}(\mathtt{CU})$

Friday October 22

Recall the definitions: The set of RNA strands S is defined (recursively) by:

Basis Step: $\quad A \in S, C \in S, U \in S, G \in S$

Recursive Step: $\quad$ If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

The function *rnalen* that computes the length of RNA strands in S is defined recursively by:

$$rnalen : S \rightarrow \mathbb{Z}^+$$

Basis Step: $\quad$ If $b \in B$ then $\quad rnalen(b) = 1$

Recursive Step: $\quad$ If $s \in S$ and $b \in B$, then $\quad rnalen(sb) = 1 + rnalen(s)$

The function *basecount* that computes the number of a given base b appearing in a RNA strand s is defined recursively by:

$$basecount : S \times B \rightarrow \mathbb{N}$$

Basis Step: $\quad$ If $b_1 \in B, b_2 \in B \quad basecount(\ (b_1, b_2)\ ) = \begin{cases} 1 & \text{when } b_1 = b_2 \\ 0 & \text{when } b_1 \neq b_2 \end{cases}$

Recursive Step: $\quad$ If $s \in S, b_1 \in B, b_2 \in B \quad basecount(\ (sb_1, b_2)\ ) = \begin{cases} 1 + basecount(\ (s, b_2)\ ) & \text{when } b_1 = b_2 \\ basecount(\ (s, b_2)\ ) & \text{when } b_1 \neq b_2 \end{cases}$

Using functions to define predicates:

L with domain $S \times \mathbb{Z}^+$ is defined by, for $s \in S$ and $n \in \mathbb{Z}^+$,

$$L(\ (s, n)\ ) = \begin{cases} T & \text{if } rnalen(s) = n \\ F & \text{otherwise} \end{cases}$$

In other words, $L(\ (s, n)\ )$ means $rnalen(s) = n$

BC with domain $S \times B \times \mathbb{N}$ is defined by, for $s \in S$ and $b \in B$ and $n \in \mathbb{N}$,

$$BC(\ (s, b, n)\ ) = \begin{cases} T & \text{if } basecount(\ (s, b)\ ) = n \\ F & \text{otherwise} \end{cases}$$

In other words, $BC(\ (s, b, n)\ )$ means $basecount(\ (s, b)\ ) = n$

Example where L evaluates to T: _____ Why?

Example where BC evaluates to T: _____ Why?

Example where L evaluates to F: _____ Why?

Example where BC evaluates to F: _____ Why?

$$\exists t \; BC(t) \qquad \exists (s,b,n) \in S \times B \times \mathbb{N} \; (basecount(\,(s,b)\,) = n)$$

In English:

Witness that proves this existential quantification is true:

$$\forall t \; BC(t) \qquad \forall (s,b,n) \in S \times B \times \mathbb{N} \; (basecount(\,(s,b)\,) = n)$$

In English:

Counterexample that proves this universal quantification is false:

New predicates from old

1. Define the **new** predicate with domain $S \times B$ and rule

$$basecount(\,(s,b)\,) = 3$$

   Example domain element where predicate is T:

2. Define the **new** predicate with domain $S \times \mathbb{N}$ and rule

$$basecount(\,(s,\texttt{A})\,) = n$$

   Example domain element where predicate is T:

3. Define the **new** predicate with domain $S \times B$ and rule

$$\exists n \in \mathbb{N} \; (basecount(\,(s,b)\,) = n)$$

   Example domain element where predicate is T:

4. Define the **new** predicate with domain S and rule

$$\forall b \in B \; (basecount(\,(s,b)\,) = 1)$$

   Example domain element where predicate is T:

Notation: for a predicate P with domain $X_1 \times \cdots \times X_n$ and a n-tuple $(x_1, \ldots, x_n)$ with each $x_i \in X$, we can write $P(x_1, \ldots, x_n)$ to mean $P(\,(x_1, \ldots, x_n)\,)$.

Nested quantifiers

$$\forall s \in S \ \forall b \in B \ \forall n \in \mathbb{N} \ (basecount(\ (s,b)\ ) = n)$$

In English:

Counterexample that proves this universal quantification is false:

$$\forall n \in \mathbb{N} \ \forall s \in S \ \forall b \in B \ (basecount(\ (s,b)\ ) = n)$$

In English:

Counterexample that proves this universal quantification is false:

Alternating nested quantifiers

$$\forall s \in S \ \exists b \in B \ (\ basecount(\ (s,b)\ ) = 3\ )$$

In English: For each RNA strand there is a base that occurs 3 times in this strand.

Write the negation and use De Morgan's law to find a logically equivalent version where the negation is applied only to the BC predicate (not next to a quantifier).

Is the original statement **True** or **False**?

$$\exists s \in S \ \forall b \in B \ \exists n \in \mathbb{N} \ (\ basecount(\ (s,b)\ ) = n\ )$$

In English: There is an RNA strand so that for each base there is some nonnegative integer that counts the number of occurrences of that base in this strand.

Write the negation and use De Morgan's law to find a logically equivalent version where the negation is applied only to the BC predicate (not next to a quantifier).

Is the original statement **True** or **False**?

Review

1.

Recall the predicate L with domain $S \times \mathbb{Z}^+$ from class, $L(\ (s,n)\ )$ means $rnalen(s) = n$. Which of the following is true? (Select all and only that apply.)

(a) $\exists s \in S\ \exists n \in \mathbb{Z}^+\ L(\ (s,n)\ )$

(b) $\exists s \in S\ \forall n \in \mathbb{Z}^+\ L(\ (s,n)\ )$

(c) $\forall n \in \mathbb{Z}^+\ \exists s \in S\ L(\ (s,n)\ )$

(d) $\forall s \in S\ \exists n \in \mathbb{Z}^+\ L(\ (s,n)\ )$

(e) $\exists n \in \mathbb{Z}^+\ \forall s \in S\ L(\ (s,n)\ )$

2.

Recall the predicate BC with domain $S \times B \times \mathbb{N}$ from class, $BC(\ (s,b,n)\ )$ means $basecount(\ (s,b)\ ) = n$. Match each sentence to its English translation, or select none of the above.

(a) $\forall s \in S\ \exists n \in \mathbb{N}\ \forall b \in B\ basecount(\ (s,b)\ ) = n$

(b) $\forall s \in S\ \forall b \in B\ \exists n \in \mathbb{N}\ basecount(\ (s,b)\ ) = n$

(c) $\forall s \in S\ \forall n \in \mathbb{N}\ \exists b \in B\ basecount(\ (s,b)\ ) = n$

(d) $\forall b \in B\ \forall n \in \mathbb{N}\ \exists s \in S\ basecount(\ (s,b)\ ) = n$

(e) $\forall n \in \mathbb{N}\ \forall b \in B\ \exists s \in S\ basecount(\ (s,b)\ ) = n$

   i. For each RNA strand and each possible base, the number of that base in that strand is a nonnegative integer.

   ii. For each RNA strand and each nonnegative integer, there is a base that occurs this many times in this strand.

   iii. Every RNA strand has the same number of each base, and that number is a nonnegative integer.

   iv. For every given nonnegative integer, there is a strand where each possible base appears the given number of times.

   v. For every given base and nonnegative integer, there is an RNA strand that has this base occurring this many times.

Challenge: Express symbolically

There are (at least) two different RNA strands that have the same number of As.

Monday October 25

Proof strategies

We now have propositional and predicate logic that can help us express statements about any domain. We will develop proof strategies to craft valid argument for proving that such statements are true or disproving them (by showing they are false). We will practice these strategies with statements about sets and numbers, both because they are familiar and because they can be used to build cryptographic systems. Then we will apply proof strategies more broadly to prove statements about data structures and machine learning applications.

When a predicate $P(x)$ is over a **finite** domain:

- To show that $\forall x P(x)$ is true: check that $P(x)$ evaluates to T at each domain element by evaluating over and over.

- To show that $\forall x P(x)$ is false: find one counterexample, a domain element where $P(x)$ evaluates to F.

- To show that $\exists x P(x)$ is true: find one witness, a domain element where $P(x)$ evaluates to T.

- To show that $\exists x P(x)$ is false: check that $P(x)$ evaluates to F at each domain element by evaluating over and over.

New! **Proof of universal by exhaustion**: To prove that $\forall x\, P(x)$ is true when P has a finite domain, evaluate the predicate at **each** domain element to confirm that it is always T.

New! **Proof by universal generalization**: To prove that $\forall x\, P(x)$ is true, we can take an arbitrary element e from the domain of quantification and show that $P(e)$ is true, without making any assumptions about e other than that it comes from the domain.
An **arbitrary** element of a set or domain is a fixed but unknown element from that set.

Definitions:

A **set** is an unordered collection of elements. When A and B are sets, $A = B$ (set equality) means

$$\forall x (x \in A \leftrightarrow x \in B)$$

When A and B are sets, $A \subseteq B$ ("A is a **subset** of B") means

$$\forall x (x \in A \rightarrow x \in B)$$

When A and B are sets, $A \subsetneq B$ ("A is a **proper subset** of B") means

$$(A \subseteq B) \wedge (A \neq B)$$

New! Proof of conditional by direct proof: To prove that the conditional statement $p \rightarrow q$ is true, we can assume p is true and use that assumption to show q is true.

New! Proof of conditional by contrapositive proof: To prove that the implication $p \rightarrow q$ is true, we can assume q is false and use that assumption to show p is also false.

New! Proof of disjuction using equivalent conditional: To prove that the disjunction $p \vee q$ is true, we can rewrite it equivalently as $\neg p \rightarrow q$ and then use direct proof or contrapositive proof.

New! Proof by Cases: To prove q, we can work by cases by first describing all possible cases we might be in and then showing that each one guarantees q. Formally, if we know that $p_1 \vee p_2$ is true, and we can show that $(p_1 \rightarrow q)$ is true and we can show that $(p_2 \rightarrow q)$, then we can conclude q is true.

New! Proof of conjunctions with subgoals: To show that $p \wedge q$ is true, we have two subgoals: subgoal (1) prove p is true; and, subgoal (2) prove q is true.

To show that $p \wedge q$ is false, it's enough to prove that $\neg p$.
To show that $p \wedge q$ is false, it's enough to prove that $\neg q$.

To prove that one set is a subset of another, e.g. to show $A \subseteq B$:

To prove that two sets are equal, e.g. to show $A = B$:

Example: $\{43, 7, 9\} = \{7, 43, 9, 7\}$

Prove or **disprove**: $\{\mathtt{A}, \mathtt{C}, \mathtt{U}, \mathtt{G}\} \subseteq \{\mathtt{AA}, \mathtt{AC}, \mathtt{AU}, \mathtt{AG}\}$

Prove or **disprove**: For some set B, $\emptyset \in B$.

Prove or **disprove**: For every set B, $\emptyset \in B$.

Prove or **disprove**: The empty set is a subset of every set.

Prove or **disprove**: The empty set is a proper subset of every set.

Prove or **disprove**: $\{4, 6\} \subseteq \{n \mid \exists c \in \mathbb{Z}(n = 4c)\}$

Prove or **disprove**: $\{4, 6\} \subseteq \{n \bmod 10 \mid \exists c \in \mathbb{Z}(n = 4c)\}$

Consider ..., an **arbitrary** .... **Assume** ..., we **want to show** that .... Which is what was needed, so the proof is complete □.

or, in other words:

Let ... be an **arbitrary** .... **Assume** ..., **WTS** that ... **QED**.

Review

1.

Suppose $P(x)$ is a predicate over a domain D.

(a) True or False: To translate the statement "There are at least two elements in D where the predicate P evaluates to true", we could write

$$\exists x_1 \in D \, \exists x_2 \in D \, (P(x_1) \wedge P(x_2))$$

(b) True or False: To translate the statement "There are at most two elements in D where the predicate P evaluates to true", we could write

$$\forall x_1 \in D \, \forall x_2 \in D \, \forall x_3 \in D \, (\, (\, P(x_1) \wedge P(x_2) \wedge P(x_3) \,) \to (\, x_1 = x_2 \vee x_2 = x_3 \vee x_1 = x_3 \,) \,)$$

2.

For each of the following English statements, select the correct translation, or select None.

Challenge: determine which of the statements are true and which are false.

(a) Every set is a subset of itself.
(b) Every set is an element of itself.
(c) Some set is an element of all sets.
(d) Some set is a subset of all sets.

   i. $\forall X \, \exists Y \, (X \in Y)$
  ii. $\exists X \, \forall Y \, (X \in Y)$
 iii. $\forall X \, \exists Y \, (X \subseteq Y)$
 iv. $\exists X \, \forall Y \, (X \subseteq Y)$
  v. $\forall X \, (X \in X)$
 vi. $\forall X \, (X \subseteq X)$

3. We want to hear how the term and this class are going for you. Please complete the midquarter feedback form: https://forms.gle/w3D7ifAWnD5sWwHf9

Wednesday October 27

Cartesian product: When A and B are sets,

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

Example: $\{43, 9\} \times \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} \times \emptyset =$

Union: When A and B are sets,
$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

Example: $\{43, 9\} \cup \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} \cup \emptyset =$

Intersection: When A and B are sets,

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

Example: $\{43, 9\} \cap \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} \cap \emptyset =$

Set difference: When A and B are sets,

$$A - B = \{x \mid x \in A \wedge x \notin B\}$$

Example: $\{43, 9\} - \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} - \emptyset =$

Disjoint sets: sets A and B are disjoint means $A \cap B = \emptyset$

Example: $\{43, 9\}, \{9, \mathbb{Z}\}$ are not disjoint

Example: The sets $\mathbb{Z}$ and $\emptyset$ are disjoint

Power set: When U is a set, $\mathcal{P}(U) = \{X \mid X \subseteq U\}$

Example: $\mathcal{P}(\{43, 9\}) =$

Example: $\mathcal{P}(\emptyset) =$

Let $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$

Example elements in W are:

Prove or **disprove**: $\forall A \in W \ \forall B \in W \ (A \subseteq B \ \rightarrow \ \mathcal{P}(A) \subseteq \mathcal{P}(B))$

Extra example: **Prove** or **disprove**: $\forall A \in W \ \forall B \in W \ (\mathcal{P}(A) = \mathcal{P}(B) \ \rightarrow \ A = B)$

Extra example: **Prove** or **disprove**: $\forall A \in W \ \forall B \in W \ \forall C \in W \ (A \cup B = A \cup C \ \rightarrow \ B = C)$

Review

1.

Let $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$. The statement

$$\forall A \in W \ \forall B \in W \ \forall C \in W \ (A \cup B = A \cup C \ \rightarrow \ B = C)$$

is false. Which of the following choices for A, B, C could be used to give a counterexample to this claim? (Select all and only that apply.)

(a) $A = \{1, 2, 3\}, B = \{1, 2\}, C = \{1, 3\}$

(b) $A = \{1, 2, 3\}, B = \{2\}, C = \{2\}$

(c) $A = \{\emptyset, 1, 2, 3\}, B = \{1, 2\}, C = \{1, 3\}$

(d) $A = \{1, 2, 3\}, B = \{1, 2\}, C = \{1, 4\}$

(e) $A = \{1, 2\}, B = \{2, 3\}, C = \{1, 3\}$

(f) $A = \{1, 2\}, B = \{1, 3\}, C = \{1, 3\}$

2.

Let $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$. Consider the statement

$$\forall A \in W \ \forall B \in W \ \big((\mathcal{P}(A) = \mathcal{P}(B)) \ \rightarrow \ (A = B)\big)$$

This statement is true. A proof of this statement starts with universal generalization, c onsidering arbitrary A and B in W. At this point, it remains to prove that $(\mathcal{P}(A) = \mathcal{P}(B)) \ \rightarrow \ (A = B)$ is true about these arbitrary elements. There are two ways to proceed:

First approach: By direct proof, in which we assume the hypothesis of the conditional and work to show that the conclusion follows.

Second approach: By proving the contrapositive version of the conditional instead, in which we assume the negation of the conclusion and work to show that the negation of hypothesis follows.

(a) First approach, assumption.

(b) First approach, "need to show".

(c) Second approach, assumption.

(d) Second approach, "need to show".

Pick an option from below for the assumption and "need to show" in each approach.

(i) $\forall X (X \subseteq A \leftrightarrow X \subseteq B)$ (v) $\forall x (x \in A \leftrightarrow x \in B)$

(ii) $\exists X (X \subseteq A \leftrightarrow X \subseteq B)$ (vi) $\exists x (x \in A \leftrightarrow x \in B)$

(iii) $\forall X (X \subseteq A \oplus X \subseteq B)$ (vii) $\forall x (x \in A \oplus x \in B)$

(iv) $\exists X (X \subseteq A \oplus X \subseteq B)$ (viii) $\exists x (x \in A \oplus x \in B)$

Friday October 29

Facts about numbers

1. Addition and multiplication of real numbers are each commutative and associative.

2. The product of two positive numbers is positive, of two negative numbers is positive, and of a positive and a negative number is negative.

3. The sum of two integers, the product of two integers, and the difference between two integers are each integers.

4. For every integer x there is no integer strictly between x and $x + 1$,

5. When x, y are positive integers, $xy \geq x$ and $xy \geq y$.

Factoring

Definition: When a and b are integers and a is nonzero, a **divides** b means there is an integer c such that $b = ac$.

Symbolically, $F(\ (a, b)\ ) = $ _____ and is a predicate over the domain _____

Other (synonymous) ways to say that $F(\ (a, b)\ )$ is true:

a is a **factor** of b a is a **divisor** of b b is a **multiple** of a $a|b$

When a is a positive integer and b is any integer, $a|b$ exactly when $b \bmod a = 0$

When a is a positive integer and b is any integer, $a|b$ exactly $b = a \cdot (b \textbf{ div } a)$

Translate these quantified statements by matching to English statement on right.

$\exists a \in \mathbb{Z}^{\neq 0}\ (\ F(\ (a, a)\ )\ )$	Every nonzero integer is a factor of itself.
$\exists a \in \mathbb{Z}^{\neq 0}\ (\ \neg F(\ (a, a)\ )\ )$	No nonzero integer is a factor of itself.
$\forall a \in \mathbb{Z}^{\neq 0}\ (\ F(\ (a, a)\ )\ )$	At least one nonzero integer is a factor of itself.
$\forall a \in \mathbb{Z}^{\neq 0}\ (\ \neg F(\ (a, a)\ )\ )$	Some nonzero integer is not a factor of itself.

Claim: Every nonzero integer is a factor of itself.

Proof:

Prove or **Disprove**: There is a nonzero integer that does not divide its square.

Prove or **Disprove**: Every positive factor of a positive integer is less than or equal to it.

Claim: Every nonzero integer is a factor of itself and every nonzero integer divides its square.

Definition: an integer n is **even** means that there is an integer a such that $n = 2a$; an integer n is **odd** means that there is an integer a such that $n = 2a + 1$. Equivalently, an integer n is **even** means $n \mod 2 = 0$; an integer n is **odd** means $n \mod 2 = 1$. Also, an integer is even if and only if it is not odd.

Definition: An integer p greater than 1 is called **prime** means the only positive factors of p are 1 and p. A positive integer that is greater than 1 and is not prime is called composite.

Extra examples: Use the definition to prove that 1 is not prime, 2 is prime, 3 is prime, 4 is not prime, 5 is prime, 6 is not prime, and 7 is prime.

True or False: The statement "There are three consecutive positive integers that are prime."

Hint: These numbers would be of the form $p, p + 1, p + 2$ (where p is a positive integer).

Proof: We need to show _____

True or False: The statement "There are three consecutive odd positive integers that are prime."

Hint: These numbers would be of the form $p, p + 2, p + 4$ (where p is an odd positive integer).

Proof: We need to show _____

Review

1.

Recall the predicate $F(\ (a, b)\ ) = $ "a is a factor of b" over the domain $\mathbb{Z}^{\neq 0} \times \mathbb{Z}$ we worked with in class. Consider the following quantified statements

(i) $\forall x \in \mathbb{Z}\ (F(\ (1, x)\ ))$

(ii) $\forall x \in \mathbb{Z}^{\neq 0}\ (F(\ (x, 1)\ ))$

(iii) $\exists x \in \mathbb{Z}\ (F(\ (1, x)\ ))$

(iv) $\exists x \in \mathbb{Z}^{\neq 0}\ (F(\ (x, 1)\ ))$

(v) $\forall x \in \mathbb{Z}^{\neq 0}\ \exists y \in \mathbb{Z}\ (F(\ (x, y)\ ))$

(vi) $\exists x \in \mathbb{Z}^{\neq 0}\ \forall y \in \mathbb{Z}\ (F(\ (x, y)\ ))$

(vii) $\forall y \in \mathbb{Z}\ \exists x \in \mathbb{Z}^{\neq 0}\ (F(\ (x, y)\ ))$

(viii) $\exists y \in \mathbb{Z}\ \forall x \in \mathbb{Z}^{\neq 0}\ (F(\ (x, y)\ ))$

(a) Select the statement whose translation is

"The number 1 is a factor of every integer."

or write NONE if none of (i)-(viii) work.

(b) Select the statement whose translation is

"Every integer has at least one nonzero factor."

or write NONE if none of (i)-(viii) work.

(c) Select the statement whose translation is

"There is an integer of which all nonzero integers are a factor."

or write NONE if none of (i)-(viii) work.

(d) For each statement (i)-(viii), determine if it is true or false.

2.

Which of the following formalizes the definition of the predicate $Pr(x)$ over the set of integers, and evaluates to T exactly when x is prime. (Select all and only correct options.)

(a) $\forall a \in \mathbb{Z}^{\neq 0}\ (\ (x > 1 \land a > 0) \rightarrow F(\ (a, x)\ )\ )$

(b) $\neg \exists a \in \mathbb{Z}^{\neq 0}\ (x > 1 \land (a = 1 \lor a = x) \land F(\ (a, x)\ ))$

(c) $(x > 1) \land \forall a \in \mathbb{Z}^{\neq 0}\ (\ (\ a > 0 \land F(\ (a, x)\ )\ ) \rightarrow (a = 1 \lor a = x)\ )$

(d) $(x > 1) \land \forall a \in \mathbb{Z}^{\neq 0}\ (\ (\ a > 1 \land \neg(a = x)\ ) \rightarrow \neg F(\ (a, x)\ )\ )$

Monday November 1

Today's session is on Zoom, log in with your @ucsd.edu account https://ucsd.zoom.us/j/97431852722 Meeting ID: 974 3185 2722

Recall the definitions: The set of RNA strands S is defined (recursively) by:

Basis Step:	$\mathtt{A} \in S, \mathtt{C} \in S, \mathtt{U} \in S, \mathtt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

The function *rnalen* that computes the length of RNA strands in S is defined recursively by:

$$rnalen : S \rightarrow \mathbb{Z}^+$$

Basis Step:	If $b \in B$ then	$rnalen(b)$	$= 1$
Recursive Step:	If $s \in S$ and $b \in B$, then	$rnalen(sb)$	$= 1 + rnalen(s)$

The function *basecount* that computes the number of a given base b appearing in a RNA strand s is defined recursively by:

$$basecount : S \times B \rightarrow \mathbb{N}$$

Basis Step: If $b_1 \in B, b_2 \in B$ $basecount(\ (b_1, b_2)\ ) = \begin{cases} 1 & \text{when } b_1 = b_2 \\ 0 & \text{when } b_1 \neq b_2 \end{cases}$

Recursive Step: If $s \in S, b_1 \in B, b_2 \in B$ $basecount(\ (sb_1, b_2)\ ) = \begin{cases} 1 + basecount(\ (s, b_2)\ ) & \text{when } b_1 = b_2 \\ basecount(\ (s, b_2)\ ) & \text{when } b_1 \neq b_2 \end{cases}$

At this point, we've seen the proof strategies

- A **counterexample** to prove that $\forall x P(x)$ is **false**.

- A **witness** to prove that $\exists x P(x)$ is **true**.

- **Proof of universal by exhaustion** to prove that $\forall x\ P(x)$ is true when P has a finite domain

- **Proof by universal generalization** to prove that $\forall x\ P(x)$ is true using an arbitrary element of the domain.

- To prove that $\exists x P(x)$ is **false**, write the universal statement that is logically equivalent to its negation and then prove it true using universal generalization.

- To prove that $p \wedge q$ is true, have two subgoals: subgoal (1) prove p is true; and, subgoal (2) prove q is true. To prove that $p \wedge q$ is false, it's enough to prove that p is false. To prove that $p \wedge q$ is false, it's enough to prove that q is false.

- Proof of conditional by **direct proof**

- Proof of conditional by **contrapositive proof**

- Proof of disjuction using equivalent conditional: To prove that the disjunction $p \vee q$ is true, we can rewrite it equivalently as $\neg p \rightarrow q$ and then use direct proof or contrapositive proof.

- **Proof by cases**.

Which proof strategies could be used to prove each of the following statements?

Hint: first translate the statements to English and identify the main logical structure.

$\forall s \in S \ (\ rnalen(s) > 0\ )$

$\forall b \in B \ \exists s \in S \ (\ basecount(\ (s, b)\ )\ > 0\ )$

$\forall s \in S \ \exists b \in B \ (\ basecount(\ (s, b)\ )\ > 0\ )$

$\exists s \in S \ (rnalen(s) = basecount(\ (s, \mathtt{A})\ )$

$\forall s \in S \ (rnalen(s) \geq basecount(\ (s, \mathtt{A})\ ))$

Claim $\forall s \in S \ (\ rnalen(s) > 0\ )$

Proof: Let s be an arbitrary RNA strand. By the recursive definition of S, either $s \in B$ or there is some strand s_0 and some base b such that $s = s_0 b$. We will show that the inequality holds for both cases.

    Case: Assume $s \in B$. We need to show $rnalen(s) > 0$. By the basis step in the definition of $rnalen$,

$$rnalen(s) = 1$$

which is greater than 0, as required.

    Case: Assume there is some strand s_0 and some base b such that $s = s_0 b$. We will show *(the stronger claim)* that
$$\forall u \in S \ \forall b \in B \ (\ rnalen(u) > 0 \rightarrow rnalen(ub) > 0\ )$$
Consider an arbitrary RNA strand u and an arbitrary base b, and assume towards a direct proof, that
$$rnalen(u) > 0$$
We need to show that $rnalen(ub) > 0$.

$$rnalen(ub) = 1 + rnalen(u) > 1 + 0 = 1 > 0$$

as required.

Proof by Structural Induction To prove a universal quantification over a recursively defined set:

    Basis Step: Show the statement holds for elements specified in the basis step of the definition.

    Recursive Step: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

Claim $\forall s \in S \, (rnalen(s) \geq basecount(\, (s, \mathtt{A}) \,))$:

Proof: We proceed by structural induction on the recursively defined set S.

Basis Case: We need to prove that the inequality holds for each element in the basis step of the recursive definition of S. Need to show

$$(\, rnalen(\mathtt{A}) \geq basecount(\, (\mathtt{A}, \mathtt{A}) \,) \,) \wedge (\, rnalen(\mathtt{C}) \geq basecount(\, (\mathtt{C}, \mathtt{A}) \,) \,)$$
$$\wedge (\, rnalen(\mathtt{U}) \geq basecount(\, (\mathtt{U}, \mathtt{A}) \,) \,) \wedge (\, rnalen(\mathtt{G}) \geq basecount(\, (\mathtt{G}, \mathtt{A}) \,) \,)$$

We calculate, using the definitions of $rnalen$ and $basecount$:

Recursive Case: We will prove that

$$\forall u \in S \, \forall b \in B \, (\, rnalen(u) \geq basecount(\, (u, \mathtt{A}) \,) \,) \rightarrow rnalen(ub) \geq basecount(\, (ub, \mathtt{A}) \,)$$

Consider arbitrary RNA strand u and arbitrary base b. Assume, as the **induction hypothesis**, that $rnalen(u) \geq basecount(\, (u, \mathtt{A}) \,)$. We need to show that $rnalen(ub) \geq basecount(\, (ub, \mathtt{A}) \,)$.

Using the recursive step in the definition of the function $rnalen$:

$$rnalen(ub) = 1 + rnalen(u)$$

The recursive step in the definition of the function $basecount$ has two cases. We notice that $b = \mathtt{A} \vee b \neq \mathtt{A}$ and we proceed by cases.

Case i. Assume $b = \mathtt{A}$.

Using the first case in the recursive step in the definition of the function $basecount$:

$$basecount(\, (ub, \mathtt{A}) \,) = 1 + basecount(\, (u, \mathtt{A}) \,)$$

By the **induction hypothesis**, we know that $basecount(\, (u, \mathtt{A}) \,) \leq rnalen(u)$ so:

$$basecount(\, (ub, \mathtt{A}) \,) = 1 + basecount(\, (u, \mathtt{A}) \,) \leq 1 + rnalen(u) = rnalen(ub)$$

and, thus, $basecount(\, (ub, \mathtt{A}) \,) \leq rnalen(ub)$, as required.

Case ii. Assume $b \neq \mathtt{A}$.

Using the second case in the recursive step in the definition of the function $basecount$:

$$basecount(\, (ub, \mathtt{A}) \,) = basecount(\, (u, \mathtt{A}) \,)$$

By the **induction hypothesis**, we know that $basecount(\, (u, \mathtt{A}) \,) \leq rnalen(u)$ so:

$$basecount(\, (ub, \mathtt{A}) \,) = basecount(\, (u, \mathtt{A}) \,) \leq rnalen(u) < 1 + rnalen(u) = rnalen(ub)$$

and, thus, $basecount(\, (ub, \mathtt{A}) \,) \leq rnalen(ub)$, as required.

Review

Recall the definitions of the functions *rnalen* and *basecount* from class.

1. Select all and only options that give a witness for the existential quantification

$$\exists s \in S \ (\ rnalen(s) = basecount(\ (s, \mathtt{U}) \ ) \ )$$

   (a) `A`
   (b) `UU`
   (c) `CU`
   (d) $(\mathtt{U}, 1)$
   (e) None of the above.

2. Select all and only options that give a counterexample for the universal quantification

$$\forall s \in S \ (\ rnalen(s) > basecount(\ (s, \mathtt{G}) \ ) \ )$$

   (a) `U`
   (b) `GG`
   (c) `AG`
   (d) `CUG`
   (e) None of the above.

3. Select all and only the true statements

   (a) $\forall s \in S \ \exists b \in B \ (\ rnalen(s) = basecount(\ (s, b) \ ) \ )$
   (b) $\exists s \in S \ \forall b \in B \ (\ rnalen(s) = basecount(\ (s, b) \ ) \ )$
   (c)

$$\forall s_1 \in S \ \forall s_2 \in S \ \forall b \in B \ \Big(\big(rnalen(s_1) = basecount(\ (s_1, b) \ ) \\ \land \, rnalen(s_2) = basecount(\ (s_2, b) \ ) \land rnalen(s_1) = rnalen(s_2)\big) \to s_1 = s_2\Big)$$

   (d) None of the above.

Wednesday November 3

To organize our proofs, it's useful to highlight which claims are most important for our overall goals. We use some terminology to describe different roles statements can have.

Theorem: Statement that can be shown to be true, usually an important one.

Less important theorems can be called **proposition**, **fact**, **result**, **claim**.

Lemma: A less important theorem that is useful in proving a theorem.

Corollary: A theorem that can be proved directly after another one has been proved, without needing a lot of extra work.

Invariant: A theorem that describes a property that is true about an algorithm or system no matter what inputs are used.
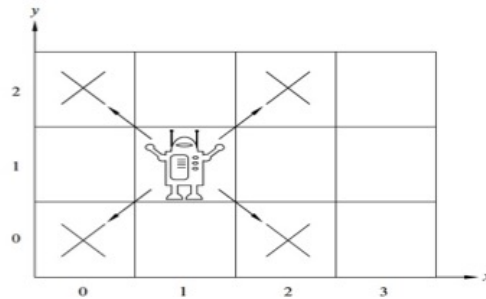
Theorem: A robot on an infinite 2-dimensional integer grid starts at $(0,0)$ and at each step moves to diagonally adjacent grid point. This robot can / cannot (*circle one*) reach $(1,0)$.

Definition The set of positions the robot can visit P is defined by:

Basis Step: $(0,0) \in P$
Recursive Step: If $(x, y) \in P$, then   are also in P

Example elements of P are:

Lemma: $\forall (x, y) \in P(x + y$ is an even integer $)$

Why are we calling this a lemma?

Proof of theorem using lemma: To show is $(1, 0) \notin P$. Rewriting the lemma to explicitly restrict the domain of the universal, we have $\forall (x, y) ((x, y) \in P \rightarrow (x + y$ is an even integer) $)$. Since the universal is true, $((1, 0) \in P \rightarrow (1 + 0$ is an even integer) $)$ is a true statement. Evaluating the conclusion of this conditional statement: By definition of long division, since $1 = 0 \cdot 2 + 1$ (where $0 \in \mathbb{Z}$ and $1 \in \mathbb{Z}$ and $0 \leq 1 < 2$ mean that 0 is the quotient and 1 is the remainder), $1 \textbf{ mod } 2 = 1$ which is not 0 so the conclusion is false. A true conditional with a false conclusion must have a false hypothesis. Thus, $(1, 0) \notin P$, QED. $\square$

Proof of lemma by structural induction:

Basis Step:

Recursive Step: Consider arbitrary $(x, y) \in P$. To show is:

$$(x + y \text{ is an even integer}) \rightarrow (\text{sum of coordinates of next position is even integer})$$

Assume **as the induction hypothesis, IH** that:

The set $\mathbb{N}$ is recursively defined. Therefore, the function $sumPow : \mathbb{N} \to \mathbb{N}$ which computes, for input i, the sum of the nonnegative powers of 2 up to and including exponent i is defined recursively by

> Basis step: $\quad sumPow(0) = 1$
> Recursive step: If $x \in \mathbb{N}$, then $\quad sumPow(x+1) = sumPow(x) + 2^{x+1}$

$sumPow(0) =$

$sumPow(1) =$

$sumPow(2) =$

Fill in the blanks in the following proof of

$$\forall n \in \mathbb{N} \ (sumPow(n) = 2^{n+1} - 1)$$

Proof: Since $\mathbb{N}$ is recursively defined, we proceed by _____.

Basis case: We need to show that _____. Evaluating each side: $LHS = sumPow(0) = 1$ by the basis case in the recursive definition of $sumPow$; $RHS = 2^{0+1} - 1 = 2^1 - 1 = 2 - 1 = 1$. Since $1 = 1$, the equality holds.

Recursive case: Consider arbitrary natural number n and assume, as the _____ that $sumPow(n) = 2^{n+1} - 1$. We need to show that _____. Evaluating each side:

$$LHS = sumPow(n+1) \overset{\text{rec def}}{=} sumPow(n) + 2^{n+1} \overset{\text{IH}}{=} (2^{n+1} - 1) + 2^{n+1}.$$

$$RHS = 2^{(n+1)+1} - 1 \overset{\text{exponent rules}}{=} 2 \cdot 2^{n+1} - 1 = (2^{n+1} + 2^{n+1}) - 1 \overset{\text{regrouping}}{=} (2^{n+1} - 1) + 2^{n+1}$$

Thus, $LHS = RHS$. The structural induction is complete and we have proved the universal generalization. $\square$

Proof by Mathematical Induction
To prove a universal quantification over the set of all integers greater than or equal to some base integer b,

> **Basis Step**: Show the property holds for b.

> **Recursive Step**: Consider an arbitrary integer n greater than or equal to b, assume (as the **induction hypothesis**) that the property holds for n, and use this and other facts to prove that the property holds for $n + 1$.

Review

1.

Recall the set P defined by the recursive definition

Basis Step:	$(0,0) \in P$
Recursive Step:	If $(x, y) \in P$ then $(x+1, y+1) \in P$ and $(x+1, y-1) \in P$ and $(x-1, y-1) \in P$ and $(x-1, y+1) \in P$

(a) Select all and only the ordered pairs below that are elements of P
   i. $(0,0)$
   ii. $(4,0)$
   iii. $(1,1)$
   iv. $(1.5, 2.5)$
   v. $(0, -2)$

(b) What is another description of the set P ? (Select all and only the true descriptions.)
   i. $\mathbb{Z} \times \mathbb{Z}$
   ii. $\{(n, n) \mid n \in \mathbb{Z}\}$
   iii. $\{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid (a+b) \bmod 2 = 0\}$

2.

Select all and only the true statements below about the relationship between structural induction and mathematical induction.

(a) Both structural induction and mathematical induction are proof strategies that may be useful when proving universal claims about recursively defined sets.

(b) Mathematical induction is a special case of structural induction, for the case when the domain of quantification is $\{n \in \mathbb{Z} \mid n \geq b\}$ for some integer b.

(c) Universal claims about the set of all integers may be proved using structural induction but not using mathematical induction.

3.

Consider the following function definitions

$$2^n : \mathbb{N} \to \mathbb{N} \text{ given by } 2^0 = 1 \quad \text{and} \quad 2^{n+1} = 2 \cdot 2^n$$

$$n! : \mathbb{N} \to \mathbb{N} \text{ given by } 0! = 1 \quad \text{and} \quad (n+1)! = (n+1)n!$$

(a) Select all and only true statements below:
  i. $2^0 < 0!$
  ii. $2^1 < 1!$
  iii. $2^2 < 2!$
  iv. $2^3 < 3!$
  v. $2^4 < 4!$
  vi. $2^5 < 5!$
  vii. $2^6 < 6!$
  viii. $2^7 < 7!$

(b) Fill in the blanks in the following proof.
  Claim: For all integers n greater than or equal to 4, $2^n < n!$

  Proof: We proceed by mathematical induction on the set of integers greater than or equal to 4.
  Basis step: Using the ____BLANK 1____,

  $$2^4 = 2 \cdot 2^3 = 2 \cdot 2 \cdot 2^2 = 2 \cdot 2 \cdot 2 \cdot 2^1 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2^0 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 1 = 16$$

  and
  $$4! = 4 \cdot 3! = 4 \cdot 3 \cdot 2! = 4 \cdot 3 \cdot 2 \cdot 1! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 24$$

  Since $16 < 24$, we have proved that $2^4 < 4!$, as required.

  Recursive step: Consider an arbitrary integer k that is greater than or equal to 4 and assume as the ____BLANK 2____, that $2^k < k!$. We want to show that $2^{k+1} < (k+1)!$.

  $$\begin{aligned}
  2^{k+1} &= 2 \cdot 2^k & \text{by } \underline{BLANK3} \\
  &< 2 \cdot k! & \text{by } \underline{BLANK4} \\
  &< k \cdot k! & \text{by } \underline{BLANK5} \\
  &< (k+1) \cdot k! & \text{by } \underline{BLANK6} \\
  &= (k+1)! & \text{by } \underline{BLANK7}
  \end{aligned}$$

  as required.
    i. properties of addition, multiplication, and $<$ for real numbers
    ii. definitions of the functions 2^n and $n!$
    iii. definition of k
    iv. induction hypothesis

Friday November 5

Definition The set of linked lists of natural numbers L is defined recursively by

$$\begin{aligned}
&\text{Basis Step:} &&[] \in L \\
&\text{Recursive Step:} &&\text{If } l \in L \text{ and } n \in \mathbb{N}, \text{ then } (n, l) \in L
\end{aligned}$$

Visually:

Example: the list with two nodes whose first node has 20 and whose second node has 42

Definition: The length of a linked list of natural numbers L, $length : L \to \mathbb{N}$ is defined by

$$\begin{aligned}
&\text{Basis Step:} &&length([\,]) &&= 0 \\
&\text{Recursive Step:} &&\text{If } l \in L \text{ and } n \in \mathbb{N}, \text{ then } &&length((n, l)) &&= 1 + length(l)
\end{aligned}$$

Definition: The function $prepend : L \times \mathbb{N} \to L$ that adds an element at the front of a linked list is defined by

Definition The function $append : L \times \mathbb{N} \to L$ that adds an element at the end of a linked list is defined by

$$\begin{aligned}
&\text{Basis Step:} &&\text{If } m \in \mathbb{N} \text{ then} \\
&\text{Recursive Step:} &&\text{If } l \in L \text{ and } n \in \mathbb{N} \text{ and } m \in \mathbb{N}, \text{ then}
\end{aligned}$$

Claim: $\forall l \in L \; (\; length(\; append(\; (l, 100)\;)\;)\;) > length(l)\;)$

Proof: By structural induction on L, we have two cases:

Basis Step

1. **To Show** $length(\; append(\; ([], 100)\;)\;)\;) > length(\; []\;)$ Because $[]$ is the only element defined in the basis step of L, we only need to prove that the property holds for $[]$.

2. **To Show** $length(\; (100, [])\;)\;) > length(\; []\;)$ By basis step in definition of $append$.

3. **To Show** $(1 + length(\; []\;)) > length(\; []\;)$ By recursive step in definition of $length$.

4. **To Show** $1 + 0 > 0$ By basis step in definition of $length$.

5. T By properties of integers

QED   Because we got to T only by rewriting **To Show** to equivalent statements, using well-defined proof techniques, and applying definitions.

Recursive Step

Consider an arbitrary: $l' \in L$, $n \in \mathbb{N}$, and we assume as the **induction hypothesis** that:

$$length(\; append(\; (l', 100\;)\;)\;)\;) > length(l')$$

Our goal is to show that $length(\; append(\; (\; (n, l'), 100\;)\;)\;)\;) > length(\; (n, l')\;)$ is also true. We start by working with one side of the candidate inequality:

$$
\begin{aligned}
LHS &= length(\; append(\; (\; (n, l'), 100\;)\;)\;) \\
&= length(\; (n, append(\; (l', 100)\;)\;)\;) &&\text{by the recursive definition of } append \\
&= 1 + length(\; append(\; (l', 100)\;)\;)\;) &&\text{by the recursive definition of } length \\
&> 1 + length(l') &&\text{by the induction hypothesis} \\
&= length((n, l')) &&\text{by the recursive definition of } length \\
&= RHS
\end{aligned}
$$

Prove or disprove: $\forall n \in \mathbb{N} \; \exists l \in L \; (\; length(l) = n \;)$

Review

Recall the definition of linked lists from class.

Consider this (incomplete) definition:

Definition The function $increment : \underline{\hspace{3cm}}$ that adds 1 to the data in each node of a linked list is defined by:

$$increment : \underline{\hspace{1.5cm}} \rightarrow \underline{\hspace{1.5cm}}$$

Basis Step: $\qquad increment([]) = []$

Recursive Step: If $l \in L, n \in \mathbb{N} \quad increment((n, l)) = (1 + n, increment(l))$

Consider this (incomplete) definition:

Definition The function $sum : L \rightarrow \mathbb{N}$ that adds together all the data in nodes of the list is defined by:

$$sum : L \rightarrow \mathbb{N}$$

Basis Step: $\qquad sum([]) = 0$

Recursive Step: If $l \in L, n \in \mathbb{N} \quad sum((n, l)) = \underline{\hspace{3cm}}$

You will compute a sample function application and then fill in the blanks for the domain and codomain of each of these functions.

1. Based on the definition, what is the result of $increment((4, (2, (7, []))))$? Write your answer directly with no spaces.

2. Which of the following describes the domain and codomain of $increment$?

   (a) The domain is L and the codomain is $\mathbb{N}$ (d) The domain is $L \times \mathbb{N}$ and the codomain is $\mathbb{N}$

   (b) The domain is L and the codomain is $\mathbb{N} \times L$ (e) The domain is L and the codomain is L

   (c) The domain is $L \times \mathbb{N}$ and the codomain is L (f) None of the above

3. Assuming we would like $sum((5, (6, [])))$ to evaluate to 11 and $sum((3, (1, (8, []))))$ to evaluate to 12, which of the following could be used to fill in the definition of the recursive case of sum?

   (a) $\begin{cases} 1 + sum(l) & \text{when } n \neq 0 \\ sum(l) & \text{when } n = 0 \end{cases}$ (c) $n + increment(l)$

   (d) $n + sum(l)$

   (b) $1 + sum(l)$ (e) None of the above

4. Choose only and all of the following statements that are **well-defined**; that is, they correctly reflect the domains and codomains of the functions and quantifiers, and respect the notational conventions we use in this class. Note that a well-defined statement may be true or false.

(a) $\forall l \in L\,(sum(l))$

(b) $\exists l \in L\,(sum(l) \wedge length(l))$

(c) $\forall l \in L\,(sum(increment(l)) = 10)$

(d) $\exists l \in L\,(sum(increment(l)) = 10)$

(e) $\forall l \in L\,\forall n \in \mathbb{N}\,((n \times l) \subseteq L)$

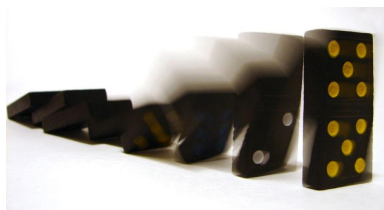(f) $\forall l_1 \in L\,\exists l_2 \in L\,(increment(sum(l_1)) = l_2)$

(g) $\forall l \in L\,(length(increment(l)) = length(l))$

5. Choose only and all of the statements in the previous part that are both well-defined and true.

Monday November 8

Visualizing induction

Proof by Mathematical Induction

To prove a universal quantification over the set of all integers greater than or equal to some base integer b,

> **Basis Step**: Show the property holds for b.

> **Recursive Step**: Consider an arbitrary integer n greater than or equal to b, assume (as the **induction hypothesis**) that the property holds for n, and use this and other facts to prove that the property holds for $n + 1$.

Proof by Strong Induction

To prove that a universal quantification over the set of all integers greater than or equal to some base integer b holds, pick a fixed nonnegative integer j and then:

Basis Step:	Show the statement holds for b, $b + 1$, ..., $b + j$.
Recursive Step:	Consider an arbitrary integer n greater than or equal to $b + j$, assume (as the **strong induction hypothesis**) that the property holds for **each of** b, $b + 1$, ..., n, and use this and other facts to prove that the property holds for $n + 1$.

Theorem: Every positive integer is a sum of (one or more) distinct powers of 2. *Binary expansions exist!*

Recall the definition for binary expansion:

Definition For n a positive integer, the **binary expansion of** n is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are each 0 or 1, $a_{k-1} \neq 0$, and

$$n = \sum_{i=0}^{k-1} a_i b^i$$

The idea in the "Least significant first" algorithm for computing binary expansions is that the binary expansion of *half* a number becomes *part* of the binary expansion of the number of itself. We can use this idea in a proof by strong induction that binary expansions exist for all positive integers n.

Proof by strong induction, with $b = 1$ and $j = 0$.

Basis step: WTS property is true about 1.

Recursive step: Consider an arbitrary integer $n \geq 1$.

Assume (as the strong induction hypothesis, IH) that the property is true about each of $1, \ldots, n$.

WTS that the property is true about $n + 1$.

Idea: We will apply the IH to $(n + 1)$ **div** 2.

Why is this ok?

Why is this helpful?

By the IH, we can write $(n+1)$ **div** 2 as a sum of powers of 2. In other words, there are values $a_{k-1}, \ldots, a_0$

such that each a_i is 0 or 1, $a_{k-1} = 1$, and

$$\sum_{i=0}^{k-1} a_i 2^i = (n+1) \textbf{ div } 2$$

Define the collection of coefficients

$$c_j = \begin{cases} a_{j-1} & \text{if } 1 \leq j \leq k \\ (n+1) \textbf{ mod } 2 & \text{if } j = 0 \end{cases}$$

Calculating:

$$\sum_{j=0}^{k} c_j 2^j = c_0 + \sum_{j=1}^{k} c_j 2^j = c_0 + \sum_{i=0}^{k-1} c_{i+1} 2^{i+1} \qquad \text{re-indexing the summation}$$

$$= c_0 + 2 \cdot \sum_{i=0}^{k-1} c_{i+1} 2^i \qquad \text{factoring out a 2 from each term in the sum}$$

$$= c_0 + 2 \cdot \sum_{i=0}^{k-1} a_i 2^i \qquad \text{by definition of } c_{i+1}$$

$$= c_0 + 2\,(\,(n+1) \textbf{ div } 2\,) \qquad \text{by IH}$$

$$= (\,(n+1) \textbf{ mod } 2\,) + 2\,(\,(n+1) \textbf{ div } 2\,) \qquad \text{by definition of } c_0$$

$$= n + 1 \qquad \text{by definition of long division}$$

Thus, $n+1$ can be expressed as a sum of powers of 2, as required.

Representing positive integers with primes

Theorem: Every positive integer *greater than 1* is a product of (one or more) primes.

Before we prove, let's try some examples:

$20 =$

$100 =$

$5 =$

Proof by strong induction, with $b = 2$ and $j = 0$.

Basis step: WTS property is true about 2.

Since 2 is itself prime, it is already written as a product of (one) prime.

Recursive step: Consider an arbitrary integer $n \geq 2$. Assume (as the strong induction hypothesis, IH) that the property is true about each of $2, \ldots, n$. WTS that the property is true about $n + 1$: We want to show that $n + 1$ can be written as a product of primes. Notice that $n + 1$ is itself prime or it is composite.

Case 1: assume $n + 1$ is prime and then immediately it is written as a product of (one) prime so we are done.

Case 2: assume that $n + 1$ is composite so there are integers x and y where $n + 1 = xy$ and each of them is between 2 and n (inclusive). Therefore, the induction hypothesis applies to each of x and y so each of these factors of $n + 1$ can be written as a product of primes. Multiplying these products together, we get a product of primes that gives $n + 1$, as required.

Since both cases give the necessary conclusion, the proof by cases for the recursive step is complete.

Sending old-fashioned mail with postage stamps

Suppose we had postage stamps worth 5 cents and 3 cents. Which number of cents can we form using these stamps? In other words, which postage can we pay?

11?

15?

4?

$$CanPay(0) \wedge \neg CanPay(1) \wedge \neg CanPay(2) \wedge$$
$$CanPay(3) \wedge \neg CanPay(4) \wedge CanPay(5) \wedge CanPay(6)$$
$$\neg CanPay(7) \wedge \forall n \in \mathbb{Z}^{\geq 8} CanPay(n)$$

where the predicate $CanPay$ with domain $\mathbb{N}$ is

$$CanPay(n) = \exists x \in \mathbb{N} \exists y \in \mathbb{N} (5x + 3y = n)$$

Proof (idea): First, explicitly give witnesses or general arguments for postages between 0 and 7. To prove the universal claim, we can use mathematical induction or strong induction.

Approach 1, mathematical induction: if we have stamps that add up to n cents, need to use them (and others) to give $n + 1$ cents. How do we get 1 cent with just 3-cent and 5-cent stamps?
Either take away a 5-cent stamps and add two 3-cent stamps,
or take away three 3-cent stamps and add two 5-cent stamps.
The details of this proof by mathematical induction are making sure we have enough stamps to use one of these approaches.

Approach 2, strong induction: assuming we know how to make postage for **all** smaller values (greater than or equal to 8), when we need to make $n+1$ cents, add one 3 cent stamp to however we make $(n + 1) - 3$ cents. The details of this proof by strong induction are making sure we stay in the domain of the universal when applying the induction hypothesis.

Review

1.

In class, we proved the theorem that: Every positive integer is a sum of (one or more) distinct powers of 2.

What's wrong with the following *attempted* proof of this fact?

Attempted proof by mathematical induction, with $b = 1$.

Basis step: WTS 1 can be written as a sum of (one or more) distinct powers of 2. Since $1 = 2^0$, we are done.

Recursive step: Consider an arbitrary integer $n \geq 1$. By the IH, we can write n as a sum of distinct powers of 2. Since $1 = 2^0$, it is a power of 2 and we can add it as a term to this sum of powers of 2. When we do so, the terms sum to $n + 1$ and we are done.

   (a) The basis step is not sufficient.

   (b) The induction hypothesis is not stated correctly.

   (c) It's wrong to say that 1 is a power of 2.

   (d) Adding the 2^0 to the sum of powers doesn't give the correct value.

   (e) Adding the 2^0 to the sum of powers is problematic for a different reason.

2.

Recall that a prime factorization is a product of primes (potentially with some of the primes occurring more than once). Select all and only the correct prime factorizations of positive integers.

   (a) $2 \cdot 2 \cdot 2 \cdot 2$

   (b) 3

   (c) $3 \cdot 4 \cdot 5$

   (d) $17 \cdot 21$

   (e) $2 \cdot 11$

3. In this question, we'll consider two possible proofs of the statement

$$\forall n \in \mathbb{Z}^{\geq 8} \exists x \in \mathbb{N} \exists y \in \mathbb{N}(5x + 3y = n)$$

(a) First approach, using mathematical induction $(b = 8)$.

Basis step: WTS property is true about 8. Consider the witnesses $x = 1$, $y = 1$. These are nonnegative integers and $5 \cdot 1 + 3 \cdot 1 = 8$, as required.

Recursive step: Consider an arbitrary $n \geq 8$. Assume (as the induction hypothesis, IH) that there are nonnegative integers x, y such that $n = 5x + 3y$. WTS that there are nonnegative integers x', y' such that $n + 1 = 5x' + 3y'$. We consider two cases, depending on whether any 5 cent stamps are used for n.

Case 1: Assume $x \geq 1$ (we assume that at least one 5 cent stamp is used for n). Define $x' = x - 1$ and $y' = y + 2$ (both in $\mathbb{N}$ by case assumption).

Calculating:

$$5x' + 3y' \overset{\text{by def}}{=} 5(x - 1) + 3(y + 2) = 5x - 5 + 3y + 6$$
$$\overset{\text{rearranging}}{=} (5x + 3y) - 5 + 6$$
$$\overset{\text{IH}}{=} n - 5 + 6 = n + 1$$

Case 2: Assume $x = 0$. Therefore $n = 3y$, so since $n \geq 8$, $y \geq 3$. Define $x' = 2$ and $y' = y - 3$ (both in $\mathbb{N}$ by case assumption). Calculating:

$$5x' + 3y' \overset{\text{by def}}{=} 5(2) + 3(y - 3) = 10 + 3y - 9$$
$$\overset{\text{rearranging}}{=} 3y + 10 - 9$$
$$\overset{\text{IH and case}}{=} n + 10 - 9 = n + 1$$

Since the goal has been proved from each case, the proof by cases is complete and we have proved the recursive step. $\square$

Why was the recursive step split into two cases?

   i. Because there are two variables x and y that need witnesses.

   ii. Because the statement has alternating quantifiers $\forall$ and $\exists$

   iii. Because the witness values need to be nonnegative and subtraction may lead to negative values.

   iv. Because the domain is all integers greater than or equal to 8.

   v. Because there are two steps in the recursive definition of $\mathbb{N}$

(b) Second approach, by strong induction $(b = 8$ and $j = 2)$

Basis step: WTS property is true about $8, 9, 10$

- Consider the witnesses $x = 1$, $y = 1$. These are nonnegative integers and $5 \cdot 1 + 3 \cdot 1 = 8$, as required.

- Consider the witnesses $x = 0$, $y = 3$. These are nonnegative integers and $5 \cdot 0 + 3 \cdot 3 = 9$, as required.

- Consider the witnesses $x = 2$, $y = 0$. These are nonnegative integers and $5 \cdot 2 + 3 \cdot 0 = 10$, as required.

Recursive step: Consider an arbitrary $n \geq 10$. Assume, as the strong induction hypothesis, that the property is true about each of $8, 9, 10, \ldots, n$. WTS that there are nonnegative integers x', y' such that $n + 1 = 5x' + 3y'$.

Since <u>Blank 1</u>, by the strong induction hypothesis, there are nonnegative integers x, y such that $(n + 1) - 3 = 5x + 3y$. Choosing <u>Blank 2</u> works because

$$5x' + 3y' = 5x + 3y + 3 = (n + 1) - 3 + 3 = n + 1.$$

Choose a true and useful statement to fill in Blank 1.

   i. $n \geq 10$ and hence $(n + 1) - 3 \geq 8$

  ii. $n \geq 8$ and hence $(n + 1) - 3 \geq 8$

 iii. $n \geq 8$ and hence $(n + 1) \geq 9$

Choose the appropriate statement to fill in Blank 2.

   i. $x' = x, y' = y$

  ii. $x' = x + 1, y' = y + 1$

 iii. $x' = x + 1, y' = y$

 iv. $x' = x, y' = y + 1$

  v. $x' = x - 1, y' = y - 1$

 vi. $x' = x - 1, y' = y$

vii. $x' = x, y' = y - 1$

Wednesday November 10

Finding a winning strategy for a game

Consider the following game: two players start with two (equal) piles of jellybeans in front of them. They take turns removing any positive integer number of jellybeans at a time from one of two piles in front of them in turns.

The player who removes the last jellybean wins the game.

Which player (if any) has a strategy to guarantee to win the game?

Try out some games, starting with 1 jellybean in each pile, then 2 jellybeans in each pile, then 3 jellybeans in each pile. Who wins in each game?

Notice that reasoning about the strategy for the 1 jellybean game is easier than about the strategy for the 2 jellybean game.

Formulate a winning strategy by working to transform the game to a simpler one we know we can win.

Player 2's Strategy: Take the same number of jellybeans that Player 1 did, but from the opposite pile.

Why is this a good idea: If Player 2 plays this strategy, at the next turn Player 1 faces a game with the same setup as the original, just with fewer jellybeans in the two piles. Then Player 2 can keep playing this strategy to win.

Claim: Player 2's strategy guarantees they will win the game.

Proof: By strong induction, we will prove that for all positive integers n, Player 2's strategy guarantees a win in the game that starts with n jellybeans in each pile.

Basis step: WTS Player 2's strategy guarantees a win when each pile starts with 1 jellybean.

In this case, Player 1 has to take the jellybean from one of the piles (because they can't take from both piles at once). Following the strategy, Player 2 takes the jellybean from the other pile, and wins because this is the last jellybean.

Recursive step: Let n be a positive integer. As the strong induction hypothesis, assume that Player 2's strategy guarantees a win in the games where there are $1, 2, \ldots, n$ many jellybeans in each pile at the start of the game.

WTS that Player 2's strategy guarantees a win in the game where there are $n + 1$ in the jellybeans in each pile at the start of the game.

In this game, the first move has Player 1 take some number, call it c (where $1 \leq c \leq n + 1$), of jellybeans from one of the piles. Playing according to their strategy, Player 2 then takes the same number of jellybeans from the other pile.

Notice that $(c = n + 1) \vee (c \leq n)$.

Case 1: Assume $c = n + 1$, then in their first move, Player 2 wins because they take all of the second pile, which includes the last jellybean.

Case 2: Assume $c \leq n$. Then after Player 2's first move, the two piles have an equal number of jellybeans. The number of jellybeans in each pile is
$$(n + 1) - c$$
and, since $1 \leq c \leq n$, this number is between 1 and n. Thus, at this stage of the game, the game appears identical to a new game where the two piles have an equal number of jellybeans between 1 and n. Thus, the strong induction hypothesis applies, and Player 2's strategy guarantees they win.

Least and greatest

For a set of numbers X, how do you formalize "there is a greatest X" or "there is a least X"?

Prove or **disprove**: There is a least prime number.

Prove or **disprove**: There is a greatest integer.

Approach 1, De Morgan's and universal generalization:

Approach 2, proof by contradiction:

Extra examples: Prove or disprove that $\mathbb{N}$, $\mathbb{Q}$ each have a least and a greatest element.

Review

1.

Recall the game Nim from class.

(a) Why did we use strong induction to prove that Player 2's strategy guarantees a win?

    i. Because there are two players in the game.

    ii. Because each turn can involve a player taking some positive number of jellybeans from a pile, not just one jellybean.

    iii. Because the strategy player 2 uses depends on what player 1 does.

    iv. Because the set of natural numbers is recursively defined.

(b) If we modify the game so that in each turn, a player could take jellybeans from one or both piles, which player has a winning strategy?

    i. Player 1.

    ii. Player 2.

    iii. Neither in general, the existence of a winning strategy for the players depends on how many jellybeans are in each pile to start.

(c) If we modify the game so that in each turn, a player must take exactly one jellybean, which player has a winning strategy?

    i. Player 1.

    ii. Player 2.

    iii. Neither in general, the existence of a winning strategy for the players depends on how many jellybeans are in each pile to start.

2.

We will prove that there is no greatest prime number

Proof Assume, towards a <u>BLANK1</u>, that there is a greatest prime number, call it n_{BIG}. In particular, this means that there are finitely many primes. Let's label them in order $p_1, \ldots, p_n$ where $p_1 = 2$ and $p_n = n_{BIG}$. Choose $r = $ <u>BLANK2</u>. We proved in class that r is **true**. It remains to show that (under our assumption) r is **false**, because that would complete the contradiction argument. Define the integer

$$C = (p_1 \cdots p_n) + 1$$

This is a positive integer greater than 1. However, we will show that it does not have any prime factors and thus is not a product of primes. By our assumption, the only prime numbers are $p_1, \ldots, p_n$. Thus, to show that C does not have any prime factors means to show that p_i is not a factor of C for each value of i from 1 to n. Towards a universal generalization, let i be an arbitrary between 1 and n (inclusive). We need to prove that p_i is not a factor of C. By definition of C,

$$C = p_i(p_1 \cdots p_{i-1} p_{i+1} \cdots p_n) + 1$$

so C **div** $p_i = p_1 \cdots p_{i-1} p_{i+1} \cdots p_n$ and C **mod** $p_i = 1$ (because $p_i > 1$ since it is prime). Since C **mod** $p_i \neq 0$, p_i is not a factor of C. Thus C witnesses that the universal claim is false, and we have proved that r is false.

(a) BLANK1

    i. universal generalization

    ii. proof of existential by witness

    iii. direct proof

    iv. proof by contrapositive

    v. proof by cases

    vi. proof by contradiction

(b) BLANK2

    i. The least prime number is 2.

    ii. There is a greatest prime number.

    iii. There is a least prime number.

    iv. Every positive integer greater than 1 is a product of primes.

    v. Every positive integer has a base expansion.

    vi. There is a greatest integer.

    vii. There is no greatest integer.

3.

Select all and only the situations in which the given proof strategy would be available.

(a) When might it be appropriate to use induction?

    i. To prove that an existential claim over the set of integers is true.

    ii. To prove that a universal claim over the real numbers is true.

    iii. To prove that a conditional claim is true.

    iv. None of the above.

(b) When might it be appropriate to use proof by contradiction?

    i. To prove that an existential claim over the set of integers is true.

    ii. To prove that a universal claim over the real numbers is true.

    iii. To prove that a conditional claim is true.

    iv. None of the above.

Friday November 12

Definition: **Greatest common divisor** Let a and b be integers, not both zero. The largest integer d such that d is a factor of a and d is a factor of b is called the greatest common divisor of a and b and is denoted by $gcd((a, b))$.

Why do we restrict to the situation where a and b are not both zero?

Calculate $gcd((10, 15))$

Calculate $gcd((10, 20))$

Claim: For any integers a, b (not both zero), $gcd((a, b)) \geq 1$.

Proof: *Show that 1 is a common factor of any two integers, so since the gcd is the greatest common factor it is greater than or equal to any common factor.*

Claim: For any positive integers a, b, $gcd((a, b)) \leq a$ and $gcd((a, b)) \leq b$.

Proof *Using the definition of gcd and the fact that factors of a positive integer are less than or equal to that integer.*

Claim: For any positive integers a, b, if a divides b then $gcd(\ (a, b)\ ) = a$.

Proof *Using previous claim and definition of gcd.*

Claim: For any positive integers a, b, c, if there is some integer q such that $a = bq + c$,

$$gcd(\ (a, b)\ ) = gcd(\ (b, c)\ )$$

Proof *Prove that any common divisor of a, b divides c and that any common divisor of b, c divides a.*

Lemma: For any integers p, q (not both zero), $gcd\left(\ \left(\ \frac{p}{gcd(\ (p,q)\ )}, \frac{q}{gcd(\ (p,q)\ )}\ \right)\ \right) = 1$. In other words, can reduce to relatively prime integers by dividing by gcd.

Proof:

Let x be arbitrary positive integer and assume that x is a factor of each of $\frac{p}{gcd(\ (p,q)\ )}$ and $\frac{q}{gcd(\ (p,q)\ )}$. This gives integers α, β such that

$$\alpha x = \frac{p}{gcd(\ (p, q)\ )} \qquad \beta x = \frac{q}{gcd(\ (p, q)\ )}$$

Multiplying both sides by the denominator in the RHS:

$$\alpha x \cdot gcd(\ (p, q)\ ) = p \qquad \beta x \cdot gcd(\ (p, q)\ ) = q$$

In other words, $x \cdot gcd(\ (p, q)\ )$ is a common divisor of p, q. By definition of gcd, this means

$$x \cdot gcd(\ (p, q)\ ) \leq gcd(\ (p, q)\ )$$

and since $gcd(\ (p, q)\ )$ is positive, this means, $x \leq 1$.

Sets of numbers

We've seen multiple representations of the set of positive integers (using base expansions and using prime factorization). Now we're going to expand our attention to other sets of numbers as well.

The **set of rational numbers**, $\mathbb{Q}$ is defined as

$$\left\{ \frac{p}{q} \mid p \in \mathbb{Z} \text{ and } q \in \mathbb{Z} \text{ and } q \neq 0 \right\} \quad \text{or, equivalently,} \quad \{x \in \mathbb{R} \mid \exists p \in \mathbb{Z} \exists q \in \mathbb{Z}^+ (p = x \cdot q)\}$$

Extra practice: Use the definition of set equality to prove that the definitions above give the same set.

We have the following subset relationships between sets of numbers:

$$\mathbb{Z}^+ \subsetneq \mathbb{N} \subsetneq \mathbb{Z} \subsetneq \mathbb{Q} \subsetneq \mathbb{R}$$

Which of the proper subset inclusions above can you prove?

Goal: The square root of 2 is not a rational number. In other words: $\neg \exists x \in \mathbb{Q}(x^2 - 2 = 0)$

Attempted proof: The definition of the set of rational numbers is the collection of fractions p/q where p is an integer and q is a nonzero integer. Looking for a **witness** p and q, we can write the square root of 2 as the fraction $\sqrt{2}/1$, where 1 is a nonzero integer. Since the numerator is not in the domain, this witness is not allowed, and we have shown that the square root of 2 is not a fraction of integers (with nonzero denominator). Thus, the square root of 2 is not rational.

The problem in the above attempted proof is that _____

Lemma 1: For every two integers a and b, not both zero, with $gcd(\ (a,b)\ ) = 1$, it is not the case that both a is even and b is even.

Lemma 2: For every integer x, x is even if and only if x^2 is even.

Proof: Towards a proof by contradiction, we will define a statement r such that $\sqrt{2} \in \mathbb{Q} \to (r \wedge \neg r)$.

Assume that $\sqrt{2} \in \mathbb{Q}$. Namely, there are positive integers p, q such that

$$\sqrt{2} = \frac{p}{q}$$

Let $a = \frac{p}{gcd(\ (p,q)\ )}$, $b = \frac{q}{gcd(\ (p,q)\ )}$, then

$$\sqrt{2} = \frac{a}{b} \qquad \text{and} \qquad gcd(\ (a,b)\ ) = 1$$

By Lemma 1, a and b are not both even. We define r to be the statement "a is even and b is even", and we have proved $\neg r$.

Squaring both sides and clearing denominator: $2b^2 = a^2$.

By definition of even, since b^2 is an integer, a^2 is even.

By Lemma 2, this guarantees that a is even too. So, by definition of even, there is some integer (call it c), such that $a = 2c$.

Plugging into the equation:
$$2b^2 = a^2 = (2c)^2 = 4c^2$$

and dividing both sides by 2
$$b^2 = 2c^2$$

and since c^2 is an integer, b^2 is even. By Lemma 2, b is even too. Thus, a is even and b is even and we have proved r.

In other words, assuming that $\sqrt{2} \in \mathbb{Q}$ guarantees $r \wedge \neg r$, which is impossible, so $\sqrt{2} \notin \mathbb{Q}$. QED

Review

1.

We will consider two ways for calculating the gcd. In each part of the question, you'll calculate $gcd((306, 120))$.

(a) The first approach uses some of the claims we proved in class to get the following algorithm:

<div align="center">Euclidean algorithm for calculating greatest common divisor</div>

```
1    procedure Euclidean(a: a positive integer, b: a positive integer)
2      x := a
3      y := b
4      while y ≠ 0
5        r := x mod y
6        x := y
7        y := r
8      return x {the result of gcd( (a,b) )}
```

Tracing this algorithm, lines 2 and 3 initialize the variables

$$x := 306 \qquad y := 120$$

Entering the while loop, the variable r is initialized to

$$r := 66$$

because $306 = 2 \cdot 120 + 66$ so $306 \bmod 120 = 66$. Calculate and fill in the updated value of r in each subsequent iteration of the **while** loop, and then give the value of $gcd((306, 120))$.

(b) The second approach uses the representation of positive integers greater than 1 as products of primes. To calculate $gcd((a, b))$ we find the prime factorizations of each of a and b, and then calculate the number that results from multiplying together terms p^c where p is a prime that appears in *both* prime factorizations of a and b and c is the *minimum* number of times p appears in the two factorizations.

Select the prime factorizations for 306 and 120 and express their gcd as a product of powers of primes.

Possible factorizations:

   i. $306 = 2 \cdot 153$, $120 = 2 \cdot 60$

  ii. $306 = 1 \cdot 2 \cdot 3 \cdot 3 \cdot 17$, $120 = 1 \cdot 3 \cdot 5 \cdot 8$

 iii. $306 = 2 \cdot 3 \cdot 3 \cdot 17$, $120 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 5$

Possible gcd choices:

   i. 2

  ii. $2 \cdot 3$

 iii. $5 \cdot 17$

 iv. $2^3 \cdot 3^2$

  v. $2^3 \cdot 3^2 \cdot 5 \cdot 8 \cdot 17$

2.

Goals for this question: recognize that we can prove the same statement in different ways. Trace proofs and justify why they are valid.

Below are two proofs of the same statement: fill in the blanks with the expressions below.

Claimed statement: **(a)**_____

> **Proof 1**: Using De Morgan's law for quantifiers, we can rewrite this statement as a universal of the negation of the body of the statement. Towards a proof by universal generalization, let x be an arbitrary element of $\mathbb{Z}$. Then we need to show that
>
> $$\textbf{(b)}\text{_____}$$
>
> We proceed by contradiction to show that
>
> $$(x \text{ is odd} \wedge x^2 \text{ is even}) \to \textbf{(c)}\text{_____}$$
>
> We assume by direct proof that $(x \text{ is odd} \wedge x^2 \text{ is even})$. Then, $(x^2 \text{ is even})$ follows directly from this assumption, so by definition of conjunction, we must show that $(x^2 \text{ is not even})$ to complete the proof. From the assumption, we have that $(x \text{ is odd})$. Applying the definition of odd, $x = 2k + 1$ for some $k \in \mathbb{Z}$. Then $x^2 = 4k^2 + 4k + 1$. We can rewrite the right hand side to $2(2k^2 + 2k) + 1$. This shows that x^2 is odd by the definition of odd, since choosing $j = 2k^2 + 2k$ gives us $j \in \mathbb{Z}$ with $x^2 = 2j + 1$. Since a number is either even or odd and not both, and x^2 is odd, then it must not be even. This concludes the proof, as we have assumed the negation of the original statement and deduced a contradiction from this assumption.

Proof 2:

1. **To Show** $\forall x \in \mathbb{Z} \neg(x$ is odd $\wedge x^2$ is even$)$ — Rewriting statement using De Morgan's law for quantifiers

2. **Choose arbitrary** $x \in \mathbb{Z}$
   To Show (d)_____ — By **(e)**_____

3. **To Show** x is odd $\rightarrow \neg(x^2$ is even$)$ — Rewrite previous **To Show** using logical equivalence

4. **Assume** x is odd
   To Show $\neg(x^2$ is even$)$ — By **(f)**_____

5. **To Show** x^2 is odd — Rewrite previous **To Show** using definition of even, odd

6. **Use the witness** k, an integer, where $x = 2k + 1$ — By existential definition of x being odd

7. **Choose the witness**
   $j = 2k^2 + 2k$, an integer
   To Show $x^2 = 2j + 1$ — Show this new **To Show** is true to prove the existential definition of x^2 being odd

8. **To Show** $(2k + 1)^2 = 2j + 1$ — Rewrite previous **To Show** using definition of k

9. **To Show** $(2k + 1)^2 = 2(2k^2 + 2k) + 1$ — Rewrite previous **To Show** using definition of j

10. **To Show** T — By algebra: multiplying out the LHS; factoring the RHS

QED — Because we got to T only by rewriting **To Show** to equivalent statements, using valid proof techniques and definitions.

Consider the following expressions as options to fill in the two proofs above. Give your answer as one of the numbers below for each blank a-c. You may use some numbers for more than one blank, but each letter only uses one of the expressions below.

i. $\exists x \in \mathbb{Z}\,(x$ is odd $\wedge x^2$ is even$)$

ii. $\neg \exists x \in \mathbb{Z}\,(x$ is odd $\wedge x^2$ is even$)$

iii. $\exists x \in \mathbb{Z}\,(x$ is odd $\wedge x$ is even$)$

iv. $\neg \exists x \in \mathbb{Z}\,(x$ is odd $\wedge x$ is even$)$

v. $\exists x \in \mathbb{Z}\,(x^2$ is odd $\wedge x^2$ is even$)$

vi. $\neg \exists x \in \mathbb{Z}\,(x^2$ is odd $\wedge x^2$ is even$)$

vii. $(x^2$ is even $\wedge x^2$ is not even$)$

viii. $\neg(x$ is odd $\wedge x^2$ is even$)$

ix. $(x$ is odd $\wedge x^2$ is even$)$

x. $(x$ is odd $\wedge x$ is not odd$)$

xi. $\neg(x$ is odd $\wedge x$ is not odd$)$

xii. x^2 is even

xiii. x^2 is odd

xiv. universal generalization

xv. proof by cases

xvi. direct proof

xvii. proof by contraposition

xviii. proof by contradiction

Monday November 15

We have the following subset relationships between sets of numbers:

$$\mathbb{Z}^+ \subsetneq \mathbb{N} \subsetneq \mathbb{Z} \subsetneq \mathbb{Q} \subsetneq \mathbb{R}$$

Which of the proper subset inclusions above can you prove?

Definition: A **finite** set is one whose distinct elements can be counted by a natural number.

Motivating question: when can we say one set is *bigger than* another?

Which is bigger?

- The set $\{1, 2, 3\}$ or the set $\{0, 1, 2, 3\}$?
- The set $\{0, \pi, \sqrt{2}\}$ or the set $\{\mathbb{N}, \mathbb{R}, \emptyset\}$?
- The set $\mathbb{N}$ or the set $\mathbb{R}^+$?

Which of the sets above are finite? infinite?

Key idea for cardinality: Counting distinct elements is a way of labelling elements with natural numbers. This is a function! In general, functions let us associate elements of one set with those of another. If the association is "*good*", we get a correspondence between the elements of the subsets which can relate the sizes of the sets.

Analogy: Musical chairs

People try to sit down when the music stops

Person☼ sits in Chair 1, Person☺ sits in Chair 2,

Person☹ is left standing!

What does this say about the number of chairs and the number of people?

Recall that a function is defined by its (1) domain, (2) codomain, and (3) rule assigning each element in the domain exactly one element in the codomain. The domain and codomain are nonempty sets. The rule can be depicted as a table, formula, English description, etc.

A function can *fail to be well-defined* if there is some domain element where the function rule doesn't give a unique codomain element. For example, the function rule might lead to more than one potential image, or to an image outside of the codomain.

Example: $f_A : \mathbb{R}^+ \to \mathbb{Q}$ with $f_A(x) = x$ is **not** a well-defined function because

Example: $f_B : \mathbb{Q} \to \mathbb{Z}$ with $f_B\left(\frac{p}{q}\right) = p + q$ is **not** a well-defined function because

Example: $f_C : \mathbb{Z} \to \mathbb{R}$ with $f_C(x) = \frac{x}{|x|}$ is **not** a well-defined function because

Definition : A function $f : D \to C$ is **one-to-one** (or injective) means for every a, b in the domain D, if $f(a) = f(b)$ then $a = b$.

Formally, $f : D \to C$ is one-to-one means _____ .

Informally, a function being one-to-one means "no duplicate images".

Definition: For nonempty sets A, B, we say that **the cardinality of A is no bigger than the cardinality of B**, and write $|A| \leq |B|$, to mean there is a one-to-one function with domain A and codomain B.

Also, we define $|\emptyset| \le |B|$ for all sets B.

In the analogy: The function $sitter : \{Chair1, Chair2\} \to \{Person\maltese, Person\copyright, Person\odot\}$ given by $sitter(Chair1) = Person\maltese$, $sitter(Chair2) = Person\copyright$, is one-to-one and witnesses that

$$|\{Chair1, Chair2\}| \le |\{Person\maltese, Person\copyright, Person\odot\}|$$

Let S_2 be the set of RNA strands of length 2, formally $S_2 = \{s \in S \mid rnalen(s) = 2\}$.

True or **False**: $|\{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}\}| \le |S_2|$

Why?

True or **False**: $|\{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}\} \times \{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}\}| \le |S_2|$

Why?

Definition: A function $f : D \to C$ is **onto** (or surjective) means for every b in the codomain, there is an element a in the domain with $f(a) = b$.

Formally, $f : D \to C$ is onto means _____.

Informally, a function being onto means "every potential image is an actual image".

Definition: For nonempty sets A, B, we say that **the cardinality of A is no smaller than the cardinality of B**, and write $|A| \ge |B|$, to mean there is an onto function with domain A and codomain B. Also, we define $|A| \ge |\emptyset|$ for all sets A.

In the analogy: The function $triedToSit : \{Person\maltese, Person\copyright, Person\odot\} \to \{Chair1, Chair2\}$ given by $triedToSit(Person\maltese) = Chair1$, $triedToSit(Person\copyright) = Chair2$, $triedToSit(Person\odot) = Chair2$, is onto and witnesses that

$$|\{Person\maltese, Person\copyright, Person\odot\}| \ge |\{Chair1, Chair2\}|$$

Let S_2 be the set of RNA strands of length 2.

True or **False**: $|S_2| \geq |\{\mathtt{A}, \mathtt{U}, \mathtt{G}, \mathtt{C}\}|$

Why?

True or **False**: $|S_2| \geq |\{\mathtt{A}, \mathtt{U}, \mathtt{G}, \mathtt{C}\} \times \{\mathtt{A}, \mathtt{U}, \mathtt{G}, \mathtt{C}\}|$

Why?

Definition : A function $f : D \to C$ is a **bijection** means that it is both one-to-one and onto. The **inverse** of a bijection $f : D \to C$ is the function $g : C \to D$ such that $g(b) = a$ iff $f(a) = b$.

Cardinality of sets

For nonempty sets A, B we say

> $|A| \leq |B|$ means there is a one-to-one function with domain A, codomain B
>
> $|A| \geq |B|$ means there is an onto function with domain A, codomain B
>
> $|A| = |B|$ means there is a bijection with domain A, codomain B

For all sets A, we say $|A| = |\emptyset|$, $|\emptyset| = |A|$ if and only if $A = \emptyset$.

Caution: we use familiar symbols to define cardinality, like $|\quad| \leq |\quad|$ and $|\quad| \geq |\quad|$ and $|\quad| = |\quad|$, but the meaning of these symbols depends on context. We've seen that vertical lines can mean absolute value (for real numbers), divisibility (for integers), and now sizes (for sets).

Now we see that $\leq$ and $\geq$ can mean comparing numbers or comparing sizes of sets. When the sets being compared are finite, the definitions of $|A| \leq |B|$ agree.

But, properties of numbers cannot be assumed when comparing cardinalities of infinite sets.

In a nutshell: cardinality of sets is defined via functions. This definition agrees with the usual notion of "size" for finite sets.

Review

1.

Select all and only the **finite** sets below.

(a) $X = \{a, b, c\}$

(b) $Y = \{1, 2, 3, 4, 5\}$

(c) $Z = \{10, 20, 30\}$

(d) $\emptyset$

(e) $\mathbb{Z}$

(f) $\{\emptyset\}$

(g) $\{\mathbb{Z}\}$

2.

Consider the following input-output definition tables with $X = \{a, b, c\}$ and $Y = \{1, 2, 3, 4, 5\}$ and $Z = \{10, 20, 30\}$

Table 1		Table 2		Table 3	
Input	Output	Input	Output	Input	Output
1	10	a	1	10	a
2	20	b	4	20	b
3	30	c	5	30	a

(a) Select all and only the tables that each define a well-defined function whose domain and codomain is each X, Y, or Z.

(b) Select all and only the tables that each define a well-defined function (with domain X or Y or Z and with codomain X or Y or Z) and that is one-to-one.

(c) Select all and only the tables that each define a well-defined function (with domain X or Y or Z and with codomain X or Y or Z) and that is onto.

3.

Consider the following functions:

$f : \mathbb{Z} \to \mathbb{N}$

$$f(n) = \begin{cases} 0 & \text{when } n = 0 \\ (-2 \cdot n) - 1 & \text{when } n < 0 \\ 2 \cdot n & \text{when } n > 0 \end{cases}$$

$g : \mathbb{Z} \to \mathbb{N}$

$$g(n) = \begin{cases} -1 \cdot n & \text{when } n < 0 \\ n & \text{when } n \geq 0 \end{cases}$$

$h : \mathbb{N} \to \mathbb{Z}$

$$h(n) = \begin{cases} (-2 \cdot n) + 1 & \text{when } n \text{ is even} \\ 2 \cdot n & \text{when } n \text{ is odd} \end{cases}$$

$q : \mathbb{N} \to \mathbb{Z}$

$$q(n) = \begin{cases} -1 \cdot ((n+1) \ \mathbf{div} \ 2) & \text{when } n \text{ is odd} \\ n \ \mathbf{div} \ 2 & \text{when } n \text{ is even} \end{cases}$$

(a) What is the result of $f(-3)$?

(b) What is the result of $q(f(-4))$?

   Notice we are looking at function composition here: first apply f and then apply q to the result.

(c) What is the result of $f(h(4))$?

   Notice we are looking at function composition here: first apply h and then apply f to the result.

(d) What is the result of $g(-4)$?

(e) What is the result of $g(4)$?

(f) Consider the following statements, and indicate if they are true for each of f, g, h, and q.

   i. This function is one-to-one.
   ii. This function is onto.
   iii. This function is a bijection.
   iv. This function could serve as a witness for $|\mathbb{Z}| \leq |\mathbb{N}|$.
   v. This function could serve as a witness for $|\mathbb{Z}| \geq |\mathbb{N}|$.
   vi. This function could serve as a witness for $|\mathbb{N}| \leq |\mathbb{Z}|$.
   vii. This function could serve as a witness for $|\mathbb{N}| \geq |\mathbb{Z}|$.

Wednesday November 17

Properties of cardinality

$$\forall A \ (\ |A| = |A| \ )$$
$$\forall A \ \forall B \ (\ |A| = |B| \ \rightarrow \ |B| = |A| \ )$$
$$\forall A \ \forall B \ \forall C \ (\ (|A| = |B| \ \wedge \ |B| = |C|) \ \rightarrow \ |A| = |C| \ )$$

Extra practice with proofs: Use the definitions of bijections to prove these properties.

Cantor-Schroder-Bernstein Theorem: For all nonempty sets,

$|A| = |B|$ if and only if $(|A| \leq |B|$ and $|B| \leq |A|)$ if and only if $(|A| \geq |B|$ and $|B| \geq |A|)$

To prove $|A| = |B|$, we can do any **one** of the following

- Prove there exists a bijection $f : A \rightarrow B$;
- Prove there exists a bijection $f : B \rightarrow A$;
- Prove there exists two functions $f_1 : A \rightarrow B$, $f_2 : B \rightarrow A$ where each of f_1, f_2 is one-to-one.
- Prove there exists two functions $f_1 : A \rightarrow B$, $f_2 : B \rightarrow A$ where each of f_1, f_2 is onto.

Definition: A set A is **countably infinite** means it is the same size as $\mathbb{N}$.

Natural numbers $\mathbb{N}$ *List*: 0 1 2 3 4 5 6 7 8 9 10...

$identity : \mathbb{N} \to \mathbb{N}$ with $identity(n) = n$

Claim: $identity$ is a bijection. *Proof*: Ex.   **Corollary**: $|\mathbb{N}| = |\mathbb{N}|$

Positive integers $\mathbb{Z}^+$ *List*: 1 2 3 4 5 6 7 8 9 10 11...

$positives : \mathbb{N} \to \mathbb{Z}^+$ with $positives(n) = n + 1$

Claim: $positives$ is a bijection. *Proof*: Ex.   **Corollary**: $|\mathbb{N}| = |\mathbb{Z}^+|$

Negative integers $\mathbb{Z}^-$ *List*: -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11...

$negatives : \mathbb{N} \to \mathbb{Z}^-$ with $negatives(n) = -n - 1$

Claim: $negatives$ is a bijection.   **Corollary**: $|\mathbb{N}| = |\mathbb{Z}^-|$

Proof: We need to show it is a well-defined function that is one-to-one and onto.

- Well-defined?

  Consider an arbitrary element of the domain, $n \in \mathbb{N}$. We need to show it maps to exactly one element of $\mathbb{Z}^-$.

- One-to-one?

  Consider arbitrary elements of the domain $a, b \in \mathbb{N}$. We need to show that

  $$(\ negatives(a) = negatives(b) \ ) \to (a = b)$$

- Onto?

  Consider arbitrary element of the codomain $b \in \mathbb{Z}^-$. We need witness in $\mathbb{N}$ that maps to b.

Integers $\mathbb{Z}$ *List*: 0 -1 1 -2 2 -3 3 -4 4 -5 5...

$f : \mathbb{Z} \to \mathbb{N}$ with $f(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ -2x - 1 & \text{if } x < 0 \end{cases}$

Claim: f is a bijection. *Proof*: Ex.   **Corollary**: $|\mathbb{Z}| = |\mathbb{N}|$

More examples of countably infinite sets

Claim: S is countably infinite

Similarly: The set of all strings over a specific alphabet is countably infinite.

Bijection using alphabetical-ish ordering (first order by length, then alphabetically among strings of same length) of strands

Claim: L is countably infinite

$$list : \mathbb{N} \to L$$
$$list(n) = (n, [])$$

$$toNum : L \to \mathbb{N}$$
$$toNum([]) = 0$$
$$toNum(\,(n, l)\,) = 2^n 3^{toNum}(l) \qquad \text{for } n \in \mathbb{N}, l \in L$$

Claim: $|\mathbb{Z}^+| = |\mathbb{Q}|$

One-to-one function from $\mathbb{Z}^+$ to $\mathbb{Q}$ is $f_1 : \mathbb{Z} \to \mathbb{Q}$ with $f_1(n) = n$ for all $n \in \mathbb{N}$.

$$f_2 : \mathbb{Q} \to \mathbb{Z} \times \mathbb{Z}$$
$$f_2(x) = \begin{cases} (0, 1) & \text{if } x = 0 \\ (p, q) & \text{if } x = \frac{p}{q}, \\ & \quad gcd(p, q) = 1, q > 0 \end{cases}$$

$$f_3 : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^+ \times \mathbb{Z}^+$$
$$f_3(\,(x, y)\,) = \begin{cases} (2x + 2, 2y + 2) & \text{if } x \geq 0, y \geq 0 \\ (-2x - 1, 2y + 2) & \text{if } x < 0, y \geq 0 \\ (2x + 2, -2y + 1) & \text{if } x \geq 0, y < 0 \\ (-2x - 1, -2y - 1) & \text{if } x < 0, y < 0 \end{cases}$$

$$f_4 : \mathbb{Z}^+ \times \mathbb{Z}^+ \to \mathbb{Z}^+$$
$$f_4(\,(x, y)\,) = 2^x 3^y \qquad \text{for } x, y \in \mathbb{Z}^+$$

Review

1. Consider the function $f : \mathbb{N} \to \mathbb{Z}$ given by $f(n) = \begin{cases} n \text{ \textbf{div} } 4 & \text{if } n \text{ is even} \\ -((n+1) \text{ \textbf{div} } 4) & \text{if } n \text{ is odd} \end{cases}$

   Select all and only the true statements below.

   (a) f is one-to-one

   (b) f is onto

   (c) f is a bijection

   (d) f witnesses that $|\mathbb{N}| \le |\mathbb{Z}|$

   (e) f witnesses that $|\mathbb{N}| \ge |\mathbb{Z}|$

   (f) f witnesses that $|\mathbb{N}| = |\mathbb{Z}|$

   (g) There is a one-to-one function with domain $\mathbb{N}$ and codomain $\mathbb{Z}$

   (h) There is an onto function with domain $\mathbb{N}$ and codomain $\mathbb{Z}$

   (i) There is a bijection with domain $\mathbb{N}$ and codomain $\mathbb{Z}$

   (j) $|\mathbb{N}| \le |\mathbb{Z}|$

   (k) $|\mathbb{N}| \ge |\mathbb{Z}|$

   (l) $|\mathbb{N}| = |\mathbb{Z}|$

2.

   Goals for this question: Reason through multiple nested quantifiers. Fluently use the definition and properties of the set of rationals.

   Recall the definition of the set of rational numbers, $\mathbb{Q} = \left\{ \frac{p}{q} \mid p \in \mathbb{Z} \text{ and } q \in \mathbb{Z} \text{ and } q \ne 0 \right\}$. We define the set of **irrational** numbers $\overline{\mathbb{Q}} = \mathbb{R} - \mathbb{Q} = \{ x \in \mathbb{R} \mid x \notin \mathbb{Q} \}$.

   (i) $\forall x \in \mathbb{Q} \; \forall y \in \mathbb{Q} \; \exists z \in \mathbb{Q} \; (x + y = z)$

   (ii) $\forall x \in \mathbb{Q} \; \forall y \in \mathbb{Q} \; \exists z \in \mathbb{Q} \; (x + z = y)$

   (iii) $\forall x \in \mathbb{Q} \; \forall y \in \mathbb{Q} \; \exists z \in \mathbb{Q} \; (x \cdot y = z)$

   (iv) $\forall x \in \mathbb{Q} \; \forall y \in \mathbb{Q} \; \exists z \in \mathbb{Q} \; (x \cdot z = y)$

   (v) $\forall x \in \overline{\mathbb{Q}} \; \forall y \in \overline{\mathbb{Q}} \; \exists z \in \overline{\mathbb{Q}} \; (x + y = z)$

   (vi) $\forall x \in \overline{\mathbb{Q}} \; \forall y \in \overline{\mathbb{Q}} \; \exists z \in \overline{\mathbb{Q}} \; (x + z = y)$

   (vii) $\forall x \in \overline{\mathbb{Q}} \; \forall y \in \overline{\mathbb{Q}} \; \exists z \in \overline{\mathbb{Q}} \; (x \cdot y = z)$

   (viii) $\forall x \in \overline{\mathbb{Q}} \; \forall y \in \overline{\mathbb{Q}} \; \exists z \in \overline{\mathbb{Q}} \; (x \cdot z = y)$

   (a) Which of the statements above (if any) could be **disproved** using the counterexample $x = \frac{1}{2}$, $y = \frac{3}{4}$?

   (b) Which of the statements above (if any) could be **disproved** using the counterexample $x = \sqrt{4}$, $y = \sqrt{3}$?

   (c) Which of the statements above (if any) could be **disproved** using the counterexample $x = 0$, $y = 3$?

   (d) Which of the statements above (if any) could be **disproved** using the counterexample $x = \sqrt{2}$, $y = 0$?

   (e) Which of the statements above (if any) could be **disproved** using the counterexample $x = \sqrt{2}$, $y = -\sqrt{2}$?

   Hint: we proved in class that $\sqrt{2} \notin \mathbb{Q}$. You may also use the facts that $\sqrt{3} \notin \mathbb{Q}$ and $-\sqrt{2} \notin \mathbb{Q}$.

   Bonus - not to hand in: prove these facts; that is, prove that $\sqrt{3} \notin \mathbb{Q}$ and $-\sqrt{2} \notin \mathbb{Q}$.

Friday November 19

Cardinality categories

A set A is **finite** means it is empty or it is the same size as $\{1, \ldots, n\}$ for some $n \in \mathbb{N}$.

A set A is **countably infinite** means it is the same size as $\mathbb{N}$. *Notice: all countably infinite sets are the same size as each other.*

A set A is **countable** means it is either finite or countably infinite.

A set A is **uncountable** means it is not countable.

Lemmas about countable and uncountable sets

Lemma: If A is a subset of a countable set, then it's countable.

Lemma: If A is a superset of an uncountable set, then it's uncountable.

Lemma: If A and B are countable sets, then $A \cup B$ is countable and $A \cap B$ is countable.

Lemma: If A and B are countable sets, then $A \times B$ is countable.

Generalize pairing ideas from $\mathbb{Z}^+ \times \mathbb{Z}^+$ to $\mathbb{Z}^+$

Lemma: If A is a subset of B, to show that $|A| = |B|$, it's enough to give one-to-one function from B to A or an onto function from A to B.

Are there always *bigger* sets?

Recall: When U is a set, $\mathcal{P}(U) = \{X \mid X \subseteq U\}$

Key idea: For finite sets, the power set of a set has strictly greater size than the set itself. Does this extend to infinite sets?

Definition: For two sets A, B, we use the notation $|A| < |B|$ to denote ($|A| \leq |B|$) $\wedge \neg$($|A| = |B|$).

$$
\begin{array}{lll}
\emptyset = \{\} & \mathcal{P}(\emptyset) = \{\emptyset\} & |\emptyset| < |\mathcal{P}(\emptyset)| \\
\{1\} & \mathcal{P}(\{1\}) = \{\emptyset, \{1\}\} & |\{1\}| < |\mathcal{P}(\{1\})| \\
\{1,2\} & \mathcal{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\} & |\{1,2\}| < |\mathcal{P}(\{1,2\})|
\end{array}
$$

$\mathbb{N}$ and its power set

Example elements of $\mathbb{N}$

Example elements of $\mathcal{P}(\mathbb{N})$

Claim: $|\mathbb{N}| \leq |\mathcal{P}(\mathbb{N})|$

Claim: There is an uncountable set. Example: _____

Proof: By definition of countable, since _____ is not finite, **to show** is $|\mathbb{N}| \neq |\mathcal{P}(\mathbb{N})|$.

Rewriting using the definition of cardinality, **to show** is

Towards a proof by universal generalization, consider an arbitrary function $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$.

To show: f is not a bijection. It's enough to show that f is not onto.

Rewriting using the definition of onto, **to show**:

$$\neg \forall B \in \mathcal{P}(\mathbb{N}) \; \exists a \in \mathbb{N} \; (\; f(a) = B \;)$$

. By logical equivalence, we can write this as an existential statement:

_____

In search of a witness, define the following collection of nonnegative integers:

$$D_f = \{ n \in \mathbb{N} \; | \; n \notin f(n) \}$$

. By definition of power set, since all elements of D_f are in $\mathbb{N}$, $D_f \in \mathcal{P}(\mathbb{N})$. It's enough to prove the following Lemma:

Lemma: $\forall a \in \mathbb{N} \; (\; f(a) \neq D_f \;)$.

Proof of lemma:

By the Lemma, we have proved that f is not onto, and since f was arbitrary, there are no onto functions from $\mathbb{N}$ to $\mathcal{P}(\mathbb{N})$. QED

Where does D_f come from? The idea is to build a set that would "disagree" with each of the images of f about some element.

$n \in \mathbb{N}$	$f(n) = X_n$	Is $0 \in X_n$?	Is $1 \in X_n$?	Is $2 \in X_n$?	Is $3 \in X_n$?	Is $4 \in X_n$?	...	Is $n \in D_f$?
0	$f(0) = X_0$	**Y / N**	Y / N	Y / N	Y / N	Y / N	...	**N / Y**
1	$f(1) = X_1$	Y / N	**Y / N**	Y / N	Y / N	Y / N	...	**N / Y**
2	$f(2) = X_2$	Y / N	Y / N	**Y / N**	Y / N	Y / N	...	**N / Y**
3	$f(3) = X_3$	Y / N	Y / N	Y / N	**Y / N**	Y / N	...	**N / Y**
4	$f(4) = X_4$	Y / N	Y / N	Y / N	Y / N	**Y / N**	...	**N / Y**
⋮								

Countable vs. uncountable: sets of numbers

Comparing $\mathbb{Q}$ and $\mathbb{R}$

Both $\mathbb{Q}$ and $\mathbb{R}$ have no greatest element.

Both $\mathbb{Q}$ and $\mathbb{R}$ have no least element.

The quantified statement
$$\forall x \forall y(x < y \rightarrow \exists z(x < z < y))$$
is true about both $\mathbb{Q}$ and $\mathbb{R}$.

Both $\mathbb{Q}$ and $\mathbb{R}$ are infinite. But, $\mathbb{Q}$ is countably infinite whereas $\mathbb{R}$ is uncountable.

The set of real numbers

$\mathbb{Z} \subsetneq \mathbb{Q} \subsetneq \mathbb{R}$

Order axioms (Rosen Appendix 1):

Reflexivity	$\forall a \in \mathbb{R}(a \leq a)$
Antisymmetry	$\forall a \in \mathbb{R} \, \forall b \in \mathbb{R} \, (\, (a \leq b \ \wedge \ b \leq a) \rightarrow (a = b) \,)$
Transitivity	$\forall a \in \mathbb{R} \, \forall b \in \mathbb{R} \, \forall c \in \mathbb{R} \, (\, (a \leq b \wedge b \leq c) \ \rightarrow \ (a \leq c) \,)$
Trichotomy	$\forall a \in \mathbb{R} \, \forall b \in \mathbb{R} \, (\, (a = b \ \vee \ b > a \ \vee \ a < b)$

Completeness axioms (Rosen Appendix 1):

Least upper bound	Every nonempty set of real numbers that is bounded above has a least upper bound
Nested intervals	For each sequence of intervals $[a_n, b_n]$ where, for each n, $a_n < a_{n+1} < b_{n+1} < b_n$, there is at least one real number x such that, for all n, $a_n \leq x \leq b_n$.

Each real number $r \in \mathbb{R}$ is described by a function to give better and better approximations

$$x_r : \mathbb{Z}^+ \to \{0,1\} \qquad \text{where } x_r(n) = n^{th} \text{ bit in binary expansion of } r$$

r	Binary expansion	x_r
0.1	$0.00011001\ldots$	$x_{0.1}(n) = \begin{cases} 0 & \text{if } n > 1 \text{ and } (n \bmod 4) = 2 \\ 0 & \text{if } n = 1 \text{ or if } n > 1 \text{ and } (n \bmod 4) = 3 \\ 1 & \text{if } n > 1 \text{ and } (n \bmod 4) = 0 \\ 1 & \text{if } n > 1 \text{ and } (n \bmod 4) = 1 \end{cases}$
$\sqrt{2} - 1 = 0.4142135\ldots$	$0.01101010\ldots$	Use linear approximations (tangent lines from calculus) to get algorithm for bounding error of successive operations. Define $x_{\sqrt{2}-1}(n)$ to be n^{th} bit in approximation that has error less than $2^{-(n+1)}$.

Claim: $\{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\}$ is uncountable.

Approach 1: Mimic proof that $\mathcal{P}(\mathbb{Z}^+)$ is uncountable.

Proof: By definition of countable, since $\{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\}$ is not finite, **to show** is $|\mathbb{N}| \ne |\{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\}|$.

To show is $\forall f : \mathbb{Z}^+ \to \{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\}$ (f is not a bijection) . Towards a proof by universal generalization, consider an arbitrary function $f : \mathbb{Z}^+ \to \{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\}$. **To show**: f is not a bijection. It's enough to show that f is not onto. Rewriting using the definition of onto, **to show**:

$$\exists x \in \{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\} \ \forall a \in \mathbb{N} \ (\ f(a) \ne x\ )$$

In search of a witness, define the following real number by defining its binary expansion

$$d_f = 0.b_1 b_2 b_3 \cdots$$

where $b_i = 1 - b_{ii}$ where b_{jk} is the coefficient of 2^{-k} in the binary expansion of $f(j)$. Since[7] $d_f \ne f(a)$ for any positive integer a, f is not onto.

Approach 2: Nested closed interval property

To show $f : \mathbb{N} \to \{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\}$ is not onto. **Strategy**: Build a sequence of nested closed intervals that each avoid some $f(n)$. Then the real number that is in all of the intervals can't be $f(n)$ for any n. Hence, f is not onto.

Consider the function $f : \mathbb{N} \to \{r \in \mathbb{R} \mid 0 \le r \ \wedge \ r \le 1\}$ with $f(n) = \frac{1+\sin(n)}{2}$

n	$f(n)$	Interval that avoids $f(n)$
0	0.5	
1	$0.920735\ldots$	
2	$0.954649\ldots$	
3	$0.570560\ldots$	
4	$0.121599\ldots$	
$\vdots$		

Other examples of uncountable sets

- The power set of any countably infinite set is uncountable. For example:

$$\mathcal{P}(\mathbb{N}), \mathcal{P}(\mathbb{Z}^+), \mathcal{P}(\mathbb{Z})$$

  are each uncountable.

- The closed interval $\{x \in \mathbb{R} \mid 0 \le x \le 1\}$, any other nonempty closed interval of real numbers whose endpoints are unequal, as well as the related intervals that exclude one or both of the endpoints.

- The set of all real numbers $\mathbb{R}$ is uncountable and the set of irrational real numbers $\overline{\overline{\mathbb{Q}}}$ is uncountable.

---

[7]There's a subtle imprecision in this part of the proof as presented, but it can be fixed.

Review

1.

The diagonalization argument constructs, for each function $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$, a set D_f defined as

$$D_f = \{x \in \mathbb{N} \mid x \notin f(x)\}$$

which has the property that, for all $n \in \mathbb{N}$, $f(n) \neq D_f$. Consider the following two functions with domain $\mathbb{N}$ and codomain $\mathcal{P}(\mathbb{N})$

$$f_1(x) = \{y \in \mathbb{N} \mid y \bmod 3 = x \bmod 3\}$$

$$f_2(x) = \{y \in \mathbb{N} \mid (y > 0) \wedge (x \bmod y \neq 0)\}$$

Select all and only the true statements below.

(a) $0 \in D_{f_1}$

(b) D_{f_1} is infinite

(c) D_{f_1} is uncountable

(d) $1 \in D_{f_2}$

(e) D_{f_2} is empty

(f) D_{f_2} is countably infinite

2.

Recall the definitions from previous assignments and class: The bases of RNA are elements of the set $B = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$. The set of RNA strands S is defined (recursively) by:

Basis Step: $\texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S$
Recursive Step: If $s \in S$ and $b \in B$, then $sb \in S$

For b an integer greater than 1 and n a positive integer, the **base b expansion of** n is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are nonnegative integers less than b, $a_{k-1} \neq 0$, and

$$n = a_{k-1} b^{k-1} + \cdots + a_1 b + a_0$$

For b an integer greater than 1, w a positive integer, and n a nonnegative integer with $n < b^w$, the **base b fixed-width w expansion of** n is

$$(a_{w-1} \cdots a_1 a_0)_{b,w}$$

where $a_0, a_1, \ldots, a_{w-1}$ are nonnegative integers less than b and

$$n = a_{w-1} b^{w-1} + \cdots + a_1 b + a_0$$

For b an integer greater than 1, w a positive integer, w' a positive integer, and x a real number the **base b fixed-width expansion of x with integer part width w and fractional part width w'** is

$$(a_{w-1} \cdots a_1 a_0.c_1 \cdots c_{w'})_{b,w,w'}$$

where $a_0, a_1, \ldots, a_{w-1}, c_1, \ldots, c_{w'}$ are nonnegative integers less than b and

$$x \geq a_{w-1}b^{w-1} + \cdots + a_1 b + a_0 + c_1 b^{-1} + \cdots + c_{w'}b^{-w'}$$

and

$$x < a_{w-1}b^{w-1} + \cdots + a_1 b + a_0 + c_1 b^{-1} + \cdots + (c_{w'} + 1)b^{-w'}$$

For each set below, determine if it is empty, nonempty and finite, countably infinite, or uncountable.

Challenge - not to hand in: how would you prove this?

(a) B

(b) S

(c) $\{x \in \mathbb{N} \mid x = (4102)_3\}$

(d) $\{x \in \mathbb{R} \mid x \text{ has a binary fixed-width 5 expansion}\}$

(e) $\{x \in \mathbb{R} \mid x = (0.10)_{(2,1,2)}\}$

Monday November 22

Definition: When A and B are sets, we say any subset of $A \times B$ is a **binary relation**. A relation R can also be represented as

- A function $f_{TF} : A \times B \to \{T, F\}$ where, for $a \in A$ and $b \in B$, $f_{TF}(\,(a,b)\,) = \begin{cases} T & \text{when } (a,b) \in R \\ F & \text{when } (a,b) \notin R \end{cases}$

- A function $f_{\mathcal{P}} : A \to \mathcal{P}(B)$ where, for $a \in A$, $f_{\mathcal{P}}(a) = \{b \in B \mid (a,b) \in R\}$

When A is a set, we say any subset of $A \times A$ is a (binary) **relation** on A.

For relation R on a set A, we can represent this relation as a **graph**: a collection of nodes (vertices) and edges (arrows). The nodes of the graph are the elements of A and there is an edge from a to b exactly when $(a,b) \in R$.

Example: For $A = \mathcal{P}(\mathbb{R})$, we can define the relation $EQ_{\mathbb{R}}$ on A as

$$\{(X_1, X_2) \in \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}) \mid |X_1| = |X_2|\}$$

Example: Let $R_{(\mathbf{mod}\ n)}$ be the set of all pairs of integers (a,b) such that (a **mod** $n = b$ **mod** n). Then a is **congruent to** b **mod** n means $(a,b) \in R_{(\mathbf{mod}\ n)}$. A common notation is to write this as $a \equiv b(\mathbf{mod}\ n)$.

$R_{(\mathbf{mod}\ n)}$ is a relation on the set _____

Some example elements of $R_{(\mathbf{mod}\ 4)}$ are:

A relation R on a set A is called **reflexive** means $(a, a) \in R$ for every element $a \in A$.

Informally, every element is related to itself.

Graphically, there are self-loops (edge from a node back to itself) at every node.

A relation R on a set A is called **symmetric** means $(b, a) \in R$ whenever $(a, b) \in R$, for all $a, b \in A$.

Informally, order doesn't matter for this relation.

Graphically, every edge has a paired "backwards" edge so we might as well drop the arrows and think of edges as undirected.

A relation R on a set A is called **transitive** means whenever $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$, for all $a, b, c \in A$.

Informally, chains of relations collapse.

Graphically, there's a shortcut between any endpoints of a chain of edges.

A relation R on a set A is called **antisymmetric** means $\forall a \in A \; \forall b \in A \; (\; (\; (a, b) \in R \land (b, a) \in R \;) \to a = b \;)$

Informally, the relation has directionality.

Graphically, can organize the nodes of the graph so that all non-self loop edges go up.

When the domain is $\{a, b, c, d, e, f, g, h\}$ define a relation that is **not reflexive** and is **not symmetric** and is **not transitive**.

When the domain is $\{a, b, c, d, e, f, g, h\}$ define a relation that is **not reflexive** but is **symmetric** and is **transitive**.

When the domain is $\{a, b, c, d, e, f, g, h\}$ define a relation that is **symmetric** and is **antisymmetric**.

Is the relation $EQ_\mathbb{R}$ reflexive? symmetric? transitive? antisymmetric?

Is the relation $R_{(\textbf{mod } 4)}$ reflexive? symmetric? transitive? antisymmetric?

Is the relation Sub on $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$ given by $Sub = \{(X, Y) \mid X \subseteq Y\}$ reflexive? symmetric? transitive? antisymmetric?

A relation is an **equivalence relation** means it is reflexive, symmetric, and transitive.

A relation is a **partial ordering** (or partial order) means it is reflexive, antisymmetric, and transitive.

For a partial ordering, its **Hasse diagram** is a graph whose nodes (vertices) are the elements of the domain of the binary relation and which are located such that nodes connected to nodes above them by (undirected) edges indicate that the relation holds between the lower node and the higher node. Moreover, the diagram omits self-loops and omits edges that are guaranteed by transitivity.

Draw the Hasse diagram of the partial order on the set $\{a, b, c, d, e, f, g\}$ defined as

$$\{(a,a), (b,b), (c,c), (d,d), (e,e), (f,f), (g,g),$$
$$(a,c), (a,d), (d,g), (a,g), (b,f), (b,e), (e,g), (b,g)\}$$

Summary: binary relations can be useful for organizing elements in a domain. Some binary relations have special properties that make them act like some familiar relations. Equivalence relations (reflexive, symmetric, transitive binary relations) "act like" equals. Partial orders (reflexive, antisymmetric, transitive binary relations) "act like" less than or equals to.

Review

1.

Recall that the binary relation $EQ_{\mathbb{R}}$ on $\mathcal{P}(\mathbb{R})$ is

$$\{(X_1, X_2) \in \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}) \mid |X_1| = |X_2|\}$$

and $R_{(\textbf{mod } n)}$ is the set of all pairs of integers (a, b) such that $(a \textbf{ mod } n = b \textbf{ mod } n)$.
Select all and only the correct items.

(a) $(\mathbb{Z}, \mathbb{R}) \in EQ_{\mathbb{R}}$

(b) $(0, 1) \in EQ_{\mathbb{R}}$

(c) $(\emptyset, \emptyset) \in EQ_{\mathbb{R}}$

(d) $(-1, 1) \in R_{(\textbf{mod } 2)}$

(e) $(1, -1) \in R_{(\textbf{mod } 3)}$

(f) $(4, 16, 0) \in R_{(\textbf{mod } 4)}$

2.

Consider the binary relation on $\mathbb{Z}^+$ defined by $\{(a, b) \mid \exists c \in \mathbb{Z}(b = ac)\}$. Select all and only the properties that this binary relation has.

(a) It is reflexive.

(b) It is symmetric.

(c) It is transitive.

(d) It is antisymmetric.

3.

(a) Consider the partial order on the set $\mathcal{P}(\{1, 2, 3\})$ given by the binary relation $\{(X, Y) \mid X \subseteq Y\}$

    i. How many nodes are in the Hasse diagram of this partial order?

    ii. How many edges are in the Hasse diagram of this partial order?

(b) Consider the binary relation on $\{1, 2, 4, 5, 10, 20\}$ defined by $\{(a, b) \mid \exists c \in \mathbb{Z}(b = ac)\}$.

    i. How many nodes are in the Hasse diagram of this partial order?

    ii. How many edges are in the Hasse diagram of this partial order?

Wednesday November 24

Exploring equivalence relations

A **partition** of a set A is a set of non-empty, disjoint subsets $A_1, A_2, \cdots, A_n$ such that

$$A = \bigcup_{i=1}^{n} A_i = \{x \mid \exists i (x \in A_i)\}$$

An **equivalence class** of an element $a \in A$ with respect to an equivalence relation R on the set A is the set

$$\{s \in A \mid (a, s) \in R\}$$

We write $[a]_R$ for this set, which is the equivalence class of a with respect to R.

Fact: When R is an equivalence relation on a nonempty set A, the collection of equivalence classes of R is a partition of A.

Also, given a partition P of A, the relation R_P on A given by

$$R_P = \{(x, y) \in A \times A \mid x \text{ and } y \text{ are in the same part of the partition } P\}$$

is an equivalence relation on A.

Recall: We say a is **congruent to b mod n** means $(a, b) \in R_{(\mathbf{mod}\ n)}$. A common notation is to write this as $a \equiv b (\mathbf{mod}\ n)$.

We can partition the set of integers using equivalence classes of $R_{(\mathbf{mod}\ 4)}$

$[0]_{R_{(\mathbf{mod}\ 4)}} =$

$[1]_{R_{(\mathbf{mod}\ 4)}} =$

$[2]_{R_{(\mathbf{mod}\ 4)}} =$

$[3]_{R_{(\mathbf{mod}\ 4)}} =$

$[4]_{R_{(\mathbf{mod}\ 4)}} =$

$[5]_{R_{(\mathbf{mod}\ 4)}} =$

$[-1]_{R_{(\mathbf{mod}\ 4)}} =$

$$\mathbb{Z} = [0]_{R_{(\mathbf{mod}\ 4)}} \ \cup \ [1]_{R_{(\mathbf{mod}\ 4)}} \ \cup \ [2]_{R_{(\mathbf{mod}\ 4)}} \ \cup \ [3]_{R_{(\mathbf{mod}\ 4)}}$$

Integers are useful because they can be used to encode other objects and have multiple representations. However, infinite sets are sometimes expensive to work with computationally. Reducing our attention to a *partition of the integers* based on congrunce mod n, where each part is represented by a (not too large) integer gives a useful compromise where many algebraic properties of the integers are preserved, and we also get the benefits of a finite domain. Moreover, modular arithmetic is well-suited to model any cyclic behavior.

Lemma: For $a, b \in \mathbb{Z}$ and positive integer n, $(a, b) \in R_{(\textbf{mod } n)}$ if and only if $n | a - b$.

Proof:

Application: Cycling

How many minutes past the hour are we at?   *Model with* $+15$ **mod** 60

Time:	12:00pm	12:15pm	12:30pm	12:45pm	1:00pm	1:15pm	1:30pm	1:45pm	2:00pm
"Minutes past":	0	15	30	45	0	15	30	45	0

Replace each English letter by a letter that's fifteen ahead of it in the alphabet   *Model with* $+15$ **mod** 26

Original index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Original letter:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Shifted letter:	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Shifted index:	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Modular arithmetic:

Lemma: For $a, b, c, d \in \mathbb{Z}$ and positive integer n, if $a \equiv b \ (\textbf{mod } n)$ and $c \equiv d \ (\textbf{mod } n)$ then $a + c \equiv b + d \ (\textbf{mod } n)$ and $ac \equiv bd \ (\textbf{mod } n)$. **Informally**: can bring mod "inside" and do it first, for addition and for multiplication.

$(102 + 48) \ \textbf{mod} \ 10 = $ _____

$(7 \cdot 10) \ \textbf{mod} \ 5 = $ _____

$(2^5) \ \textbf{mod} \ 3 = $ _____

Application: Cryptography

Definition: Let a be a positive integer and p be a large[8] prime number, both known to everyone. Let k_1 be a secret large number known only to person P_1 (Alice) and k_2 be a secret large number known only to person P_2 (Bob). Let the **Diffie-Helman shared key** for a, p, k_1, k_2 be $(a^{k_1 \cdot k_2} \bmod p)$.

Idea: P_1 can quickly compute the Diffie-Helman shared key knowing only a, p, k_1 and the result of $a^{k_2} \bmod p$ (that is, P_1 can compute the shared key without knowing k_2, only $a^{k_2} \bmod p$). Similarly, P_2 can quickly compute the Diffie-Helman shared key knowing only a, p, k_2 and the result of $a^{k_1} \bmod p$ (that is, P_2 can compute the shared key without knowing k_1, only $a^{k_1} \bmod p$). But, any person P_3 who knows neither k_1 nor k_2 (but may know any and all of the other values) cannot compute the shared secret efficiently.

Key property for *shared* secret:

$$\forall a \in \mathbb{Z}\, \forall b \in \mathbb{Z}\, \forall g \in \mathbb{Z}^+\, \forall n \in \mathbb{Z}^+ ((g^a \bmod n)^b, (g^b \bmod n)^a) \in R_{(\mathbf{mod}\ n)}$$

Key property for shared *secret*:

There are efficient algorithms to calculate the result of modular exponentiation but there are no (known) efficient algorithms to calculate discrete logarithm.

---

[8]We leave the definition of "large" vague here, but think hundreds of digits for practical applications. In practice, we also need a particular relationship between a and p to hold, which we leave out here.

Review

1.

Fill in the blanks in the following proof that, for any equivalence relation R on a set A,

$$\forall a \in A \; \forall b \in A \;\; ((a,b) \in R \leftrightarrow [a]_R \cap [b]_R \neq \emptyset)$$

Proof: Towards a **(a)**_____, consider arbitrary elements a, b in A. We will prove the biconditional statement by proving each direction of the conditional in turn.

Goal 1: we need to show $(a,b) \in R \rightarrow [a]_R \cap [b]_R \neq \emptyset$ *Proof of Goal 1*: Assume towards a **(b)**_____ that $(a,b) \in R$. We will work to show that $[a]_R \cap [b]_R \neq \emptyset$. Namely, we need an element that is in both equivalence classes, that is, we need to prove the existential claim $\exists x \in A \; (x \in [a]_R \wedge x \in [b]_R)$. Towards a **(c)**_____, consider $x = b$, an element of A by definition. By **(d)**_____ of R, we know that $(b,b) \in R$ and thus, $b \in [b]_R$. By assumption in this proof, we have that $(a,b) \in R$, and so by definition of equivalence classes, $b \in [a]_R$. Thus, we have proved both conjuncts and this part of the proof is complete.

Goal 2: we need to show $[a]_R \cap [b]_R \neq \emptyset \rightarrow (a,b) \in R$ *Proof of Goal 2*: Assume towards a **(e)**_____ that $[a]_R \cap [b]_R \neq \emptyset$. We will work to show that $(a,b) \in R$. By our assumption, the existential claim $\exists x \in A \; (x \in [a]_R \wedge x \in [b]_R)$ is true. Call w a witness; thus, $w \in [a]_R$ and $w \in [b]_R$. By definition of equivalence classes, $w \in [a]_R$ means $(a,w) \in R$ and $w \in [b]_R$ means $(b,w) \in R$. By **(f)**_____ of R, $(w,b) \in R$. By **(g)**_____ of R, since $(a,w) \in R$ and $(w,b) \in R$, we have that $(a,b) \in R$, as required for this part of the proof.

Consider the following expressions as options to fill in the two proofs above. Give your answer as one of the numbers below for each blank a-c. You may use some numbers for more than one blank, but each letter only uses one of the expressions below.

i exhaustive proof	vi proof by contrapositive
ii proof by universal generalization	vii proof by contradiction
iii proof of existential using a witness	viii reflexivity
iv proof by cases	ix symmetry
v direct proof	x transitivity

2.

Modular exponentiation is required to carry out the Diffie-Helman protocol for computing a shared secret over an unsecure channel.

Consider the following algorithm for fast exponentiation (based on binary expansion of the exponent).

Modular Exponentiation

```
1    procedure modular exponentiation(b: integer;
2              n = (a_{k-1}a_{k-2}...a_1a_0)_2, m: positive integers)
3    x := 1
4    power := b mod m
5    for i:= 0 to k-1
6       if a_i = 1 then x:= (x · power) mod m
7       power := (power · power) mod m
8    return x {x equals b^n mod m}
```

(a) If we wanted to calculate $3^8 \bmod 7$ using the modular exponentation algorithm above, what are the values of the parameters b, n, and m? (Write these values in usual, decimal-like, mathematical notation.)

(b) Give the output of the *modular exponentiation* algorithm with these parameters, i.e. calculate $3^8 \bmod 7$. (Write these values in usual, decimal-like, mathematical notation.)

Friday November 26

No class, in observance of Thanksgiving holiday.