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 $\sum_{i=1}^{n} x_i$	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*	$7 \in \{43, 7, 9\}$ $\quad$ $2 \notin \{43, 7, 9\}$
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 $B = \{\mathtt{A}, \mathtt{C}, \mathtt{U}, \mathtt{G}\}$.

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

where sb is string concatenation.

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 $x + 2 \in \mathbb{Z}$

Wednesday September 29

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`
	$(\texttt{A}, \texttt{C})$		$(\texttt{U}, \texttt{U})$	
$B \times \{-1, 0, 1\}$				
$\{-1, 0, 1\} \times B$				
			$(0, 0, 0)$	
$\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\} \circ \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$				
			`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 $\to$ CODOMAIN be given by
FUNCTION-NAME(x) = ... for every $x \in DOMAIN$".

or

"Consider the function FUNCTION-NAME: DOMAIN $\to$ CODOMAIN given by
FUNCTION-NAME(x) = ... 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 : $\to$
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:

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.

Definition (Of a function, recursively) 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	$rnalen(sb)$	$= 1 + rnalen(s)$

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:

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

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

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

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 $B = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$. 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 $sb \in S$

   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 $\{\texttt{AA}, \texttt{AC}\} \circ \{\texttt{U}, \texttt{AA}\}$?

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

   (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 $\{\texttt{AA}, \texttt{AC}\} \circ \{\texttt{U}, \texttt{AA}\}$ and in the set obtained as the result of the Cartesian product $\{\texttt{AA}, \texttt{AC}\} \times \{\texttt{UA}, \texttt{AA}\}$

   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: $2^0 =$ ___

   (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)

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

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 $(A)_{16} = (10)_{10}$ $(B)_{16} = (11)_{10}$ $(C)_{16} = (12)_{10}$ $(D)_{16} = (13)_{10}$ $(E)_{16} = (14)_{10}$ $(F)_{16} = (15)_{10}$

Common bases:	Binary $b = 2$	Octal $b = 8$	Decimal $b = 10$	Hexadecimal $b = 16$

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 $0 \leq r < d$, 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.

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 log_b 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 $k > 1$)

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

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

```
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) $(3)_8$