Let's get started

We want you to be successful.

We will work together to build an environment in CSE 20 that supports your learning in a way that respects your perspectives, experiences, and identities. Our goal is for you to engage with interesting and challenging concepts and feel comfortable exploring, asking questions, and thriving.

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/

Financial aid resources, the possibility of emergency grant funding, and off-campus housing referral resources are available: see your College Dean of Student Affairs.

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. Here are some campus contacts that could provide this help: Counseling and Psychological Services (CAPS) at 858 534-3755 or http://caps.ucsd.edu; OPHD at 858 534-8298 or ophd@ucsd.edu , http://ophd.ucsd.edu; CARE at Sexual Assault Resource Center at 858 534-5793 or sarc@ucsd.edu , http://care.ucsd.edu.

Please reach out (minnes@ucsd.edu) if you need support with extenuating circumstances affecting CSE 20.

Welcome to CSE 20: Discrete Math for CS in Winter 2026!

Class website: https://canvas.ucsd.edu/ Instructor: Prof. Mia Minnes "Minnes" rhymes with Guinness, minnes@ucsd.edu, http://cseweb.ucsd.edu/ minnes. CSE 20 team: One instructor + two TAs and three tutors + all of you

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

On a typical week in CSE 20: **MWF** Lectures (sometimes with pre-class reading), **M** Discussion + review quiz due, then (every second) **Th** Homework due. Office hours (hosted by instructors and TAs and tutors) where you can come to talk about course concepts and ask for help as you work through sample problems and Q+A on Piazza available throughout the week. CSE 20 has two in-term tests and one final exam this quarter. What you can find on the class website on Canvas (https://canvas.ucsd.edu/):

1. Syllabus
2. Notes for class + annotations
3. PrairieLearn + PrairieTest
4. Assignments (PDF, tex, solutions)
5. Gradescope
6. Piazza
7. Dates

There are lots of great reasons to have a laptop, tablet, or phone open during class. You might be taking notes, getting a photo of an important moment on the board, trying out a construction that we're developing together, working on the review quiz, and so on. The main issue with screens and technology in the classroom isn't that they might distract you, it's the distraction of other students. We ask that if you would like to use a device in class and may have have unrelated content open, please sit in one of the back two rows of the room so that it's not adversely affecting other students.

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

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).

Week 1 at a glance

We will be learning and practicing to:

- Model systems with tools from discrete mathematics and reason about implications of modelling choices. Explore applications in CS through multiple perspectives, including software, hardware, and theory.
  - Selecting and representing appropriate data types and using notation conventions to clearly communicate choices
- Translate between different representations to illustrate a concept.
  - Translating between symbolic and English versions of statements using precise mathematical language
- Use precise notation to encode meaning and present arguments concisely and clearly
  - Defining important sets of numbers, e.g. set of integers, set of rational numbers
  - Precisely describing a set using appropriate notation e.g. roster method, set builder notation, and recursive definitions
  - Defining functions using multiple representations
- 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.
  - Using a recursive definition to evaluate a function or determine membership in a set

TODO:

#FinAid Assignment on Canvas https://canvas.ucsd.edu/courses/71479/quizzes/235977 (complete as soon as possible)

Quizzes "preclass survey"
"#FinAid survey"

Review quiz based on class material (due Monday 01/12/2026)

Homework assignment 1 (due Thursday 01/15/2026)

Week 1 Monday: Modeling applications

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

prior reactions to recommendation + prior searches (as a proxy for interest)

viewing history (as a proxy for interest) (including actors + directors)

DOB and/or maturity/ratings of movies watched (as a proxy for age)

genres + languages

demographics

device type / location + OS (for contextualized 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.

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*	$7 \in \{43, 7, 9\}$ $2 \notin \{43, 7, 9\}$ $=$ $\{43, 7, 43, 9\}$ $=$ $\{9, 43, 7\}$
n-tuple ordered sequence of elements with n "slots" ($n > 0$) *repetition matters, fixed length* 1-tuple (43) *Equal n-tuples have corresponding components equal*	2-tuples $(43, 9)$ $(43, \emptyset)$ $(9, 43)$ 3-tuple $(9, 9, 43)$
string ordered finite sequence of elements each from specified set (called the alphabet over which the string is defined) *repetition matters, arbitrary finite length* *Equal strings have same length and corresponding characters equal*	Alphabet $\{0, 1\}$ 001 11 10 Alphabet $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ AOA

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 $\emptyset$.

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).

opinion about first movie

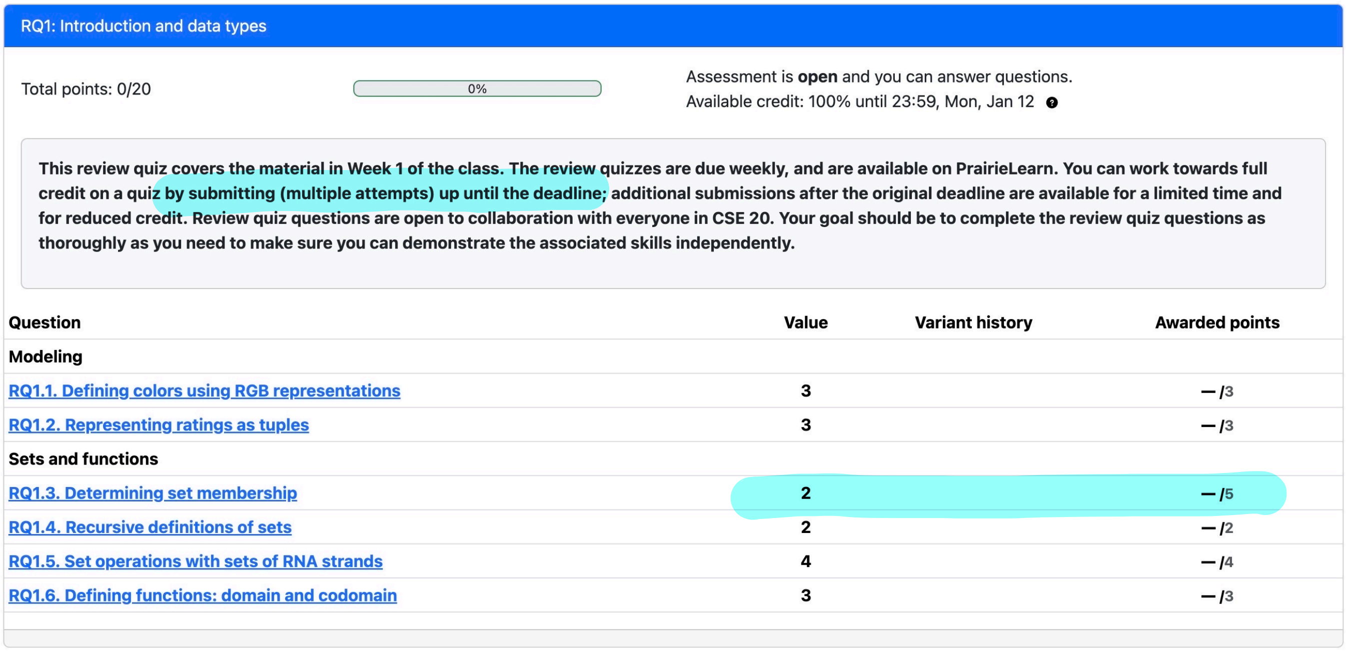
opinion about second movie

Fixed order of movies $(\underline{\qquad}, \underline{\qquad}, \underline{\qquad}, \underline{\qquad})$

4 movies!

Person	Superman	Wicked	KPop Demon Hunters	A Minecraft Movie	Ratings written as a 4-tuple
P_1	✗	•	✓	✓	$(-1, 0, 1, 1)$
P_2	✓	✓	✗	✓	$(1, 1, -1, 1)$
P_3	✓	✓	✓	✓	$(1, 1, 1, 1)$
P_4	•	✗	✓	✗	$(0, -1, 1, -1)$
You					

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



RQ1: Introduction and data types

Total points: 0/20 0%

Assessment is **open** and you can answer questions.
Available credit: 100% until 23:59, Mon, Jan 12

This review quiz covers the material in Week 1 of the class. The review quizzes are due weekly, and are available on PrairieLearn. You can work towards full credit on a quiz by submitting (multiple attempts) up until the deadline; additional submissions after the original deadline are available for a limited time and for reduced credit. Review quiz questions are open to collaboration with everyone in CSE 20. Your goal should be to complete the review quiz questions as thoroughly as you need to make sure you can demonstrate the associated skills independently.

Question	Value	Variant history	Awarded points
Modeling			
RQ1.1. Defining colors using RGB representations	3		— /3
RQ1.2. Representing ratings as tuples	3		— /3
Sets and functions			
RQ1.3. Determining set membership	2		— /5
RQ1.4. Recursive definitions of sets	2		— /2
RQ1.5. Set operations with sets of RNA strands	4		— /4
RQ1.6. Defining functions: domain and codomain	3		— /3

Week 1 Wednesday: Defining sets

LaTeX

\mathbb

Notation and prerequisites

Term	Notation Example(s)	We say in English ...
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 (nonnegative integers). *We use the convention that 0 is a natural number.*

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 }.

{ Superman, Wicked, KPop Demon Hunters, A Minecraft Movie }

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

$$\{x \in U \mid \underbrace{...x...}_{\text{property true/false of } x}\}$$

or form "the collection of all outputs of some operation when the input ranges over the universe U" by writing

$$\{\underbrace{...x...}_{\text{operation using } x} \mid x \in U\}$$

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.

Can we infer the data type of the example element from the notation?

$\{-1, 1\}$ Roster An example element is -1

$= \{1, -1\}$

$= \{x \in \mathbb{Z} \mid -1 \leq x \leq 1 \text{ and } x \neq 0\}$ Set builder

$\{0, 0\}$ Roster ~~An~~ The example element is 0

$= \{0\}$

$\{-1, 0, 1\}$ Roster An example element is 0

$\{\underbrace{(x, x, x)}_{\text{3-tuple}} \mid x \in \{-1, 0, 1\}\}$ Set builder An example element is $(0,0,0)$

$(-1, 0, 0)$ is not in this set

$\{\}$ Roster No elements in this set

$= \emptyset$

$\{x \in \mathbb{Z} \mid x \geq 0\}$ Set builder An example element is 1

$= \mathbb{N}$

$\{x \in \mathbb{Z} \mid x > 0\}$ Set builder An example element is 1

$= \mathbb{Z}^+$

$\{\smile, ☼\}$ Roster An example element is $\smile$

$\{\texttt{A}, \texttt{C}, \texttt{U}, \texttt{G}\}$ Roster An example element is C

$= B$

$\{\texttt{AUG}, \texttt{UAG}, \texttt{UGA}, \texttt{UAA}\}$ Roster An example element is UGA

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

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: $0 \in \mathbb{N}$

Recursive Step: When $x \in \mathbb{N}$, $x+1 \in \mathbb{N}$.

$\dot{0} \quad \dot{1} \quad \dot{2} \quad \dot{3} \quad \dot{4} \quad \ldots$

Examples:

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

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

Recursive Step: When $x \in \mathbb{N}$, $x+1$ in $\mathbb{N}$ and $x-1 \in \mathbb{N}$

Examples:

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

For each $b \in B$, $b \in S$

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

Recursive Step: If $\underset{\text{strand}}{s} \in S$ and $\boxed{b \in B}$, then $\underset{\text{more complicated strand is also an RNA strand.}}{sb} \in S$

simpler

$b = A$ When $s \in S$, $sA \in S$

$b = C$ When $s \in S$, $sC \in S$

$b = U$ when $s \in S$, $sU \in S$

$b = G$ When $s \in S$, $sG \in S$

where sb is string concatenation.

Examples: A (basis step) G (basis step)

$\underset{s}{A}\underset{b}{U}$ (recursive step) $\underset{s}{AU}\,\underset{b}{C}$ (recursive step)

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

Basis Step: λ is a bitstring

Recursive Step: When u is a bitstring, so is $u0$ and so is $u1$.

Notation: We call the set of bitstrings $\{0, 1\}^*$ and we say this is the set of all strings over $\{0, 1\}$.

Examples: λ $\quad 0 \quad 00 \quad 010$

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 $y \in Y$

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

$X \times Y \times Z = \{(x,y,z) \mid x \in X, y \in Y, z \in Z\}$

$X_1 \times X_2 \times X_3 \times X_4 = \{(x_1,x_2,x_3,x_4) \mid x_i \in X_i \text{ for } 1 \le i \le 4\}$

Conventions: $\boxed{(1)}$ Cartesian products can be chained together to result in sets of n-tuples and (2) When we form the Cartesian product of a set with itself $X \times X$ we can denote that set as X^2, or X^n for the Cartesian product of a set with itself n times for a positive integer n.

$\mathbb{R}^2$

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 and their data type**:
B	`A` `C` `G` `U`
$\{\texttt{A},\texttt{U}\} \times B$	$(\texttt{A},\texttt{C})$ $(\texttt{U},\texttt{U})$
$B \times \{-1, 0, 1\}$	$(A, -1)$
$\{-1, 0, 1\} \times B$	$(-1, A)$
Alternative: $\mathbb{Z}^3$ $\{-1,0,1\} \times \{0\} \times \mathbb{Z}$	$(0, 0, 0)$
$\{\texttt{A},\texttt{C},\texttt{G},\texttt{U}\} \circ \{\texttt{A},\texttt{C},\texttt{G},\texttt{U}\}$	AA GC
$\{GG, AC\} \circ \{U, GG\}$	GG|GG

Week 1 Friday: Defining functions

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
piecewise rule definition	$f(x) = \begin{cases} \text{rule 1 for } x & \text{when COND 1} \\ \text{rule 2 for } x & \text{when COND 2} \end{cases}$	Define f of x to be the result of applying rule 1 to x when condition COND 1 is true and the result of applying rule 2 to x when condition COND 2 is true. This can be generalized to having more than two conditions (or cases).
function application	$f(7)$	f of 7 **or** f applied to 7 **or** the image of 7 under f
	$f(z)$	f of z **or** f applied to z **or** the image of z under f
	$f(g(z))$	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

Pro-tip: the meaning of two vertical lines | | depends on the data-types of what's between the lines. For example, when placed around a number, the two vertical lines represent absolute value. We've seen a single vertial line | used as part of set builder definitions to represent "such that". Again, this is (one of the many reasons) why is it very important to declare the data-type of variables before we use them.

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.

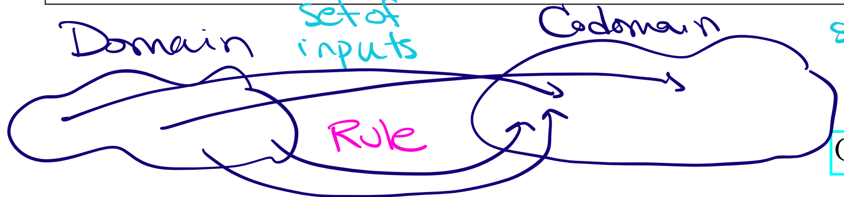
The domain and codomain are nonempty sets.
The rule can be depicted as a table, formula, piecewise definition, or English description.
The notation is

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

rule

or

"Consider the function FUNCTION-NAME: DOMAIN → CODOMAIN defined as
FUNCTION-NAME(x) = . . . for every $x \in DOMAIN$".



set of possible outputs i.e. "output type"

Example: The absolute value function

Domain $\mathbb{R}$

Codomain $\mathbb{R}$

Rule

the absolute value of x is x when $x \geq 0$ and it's $-x$ when $x < 0$.

$$|x| = \begin{cases} x, & \overset{\text{if}}{\text{when}}\ x \geq 0 \\ -x, & \overset{\text{if}}{\text{when}}\ x < 0 \end{cases}$$

Applying this function to domain elements

* when $x = 3$, the absolute value of x is 3.
* when $x = -3$, the absolute value of x is $-x = -(-3) = 3$

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 $P_1 = (-1, 0, 1, 0)$, $P_2 = (1, 1, -1, 0)$, $P_3 = (1, 1, 1, 0)$, $P_4 = (0, -1, 1, 0)$ 4-tuples

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 quantify difference among user preferences.

For example, consider the function $d_0 : \underbrace{\{-1, 0, 1\}^4 \times \{-1, 0, 1\}^4}_{\text{DOMAIN}} \to \overset{\text{CODOMAIN}}{\mathbb{R}}$ given by

Rule: $d_0(\ (\ (x_1, x_2, x_3, x_4), (y_1, y_2, y_3, y_4)\ )\ ) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + (x_4 - y_4)^2}$

Domain of d_0 is $\underbrace{\{-1,0,1\}^4 \times \{-1,0,1\}^4}_{\text{Cartesian Product}} = \{ (u,v) \mid$ u, v are each 4-tuples representing user ratings $\}$.

Codomain of d_0 is $\mathbb{R}$

Sample application

$$d_0(\ (P_1, P_2)\ ) = d_0\Big(\ \underbrace{(\overset{u=P_1}{(-1,0,1,0)}, \overset{v=P_2}{(1,1,-1,0)})}_{\text{input}}\ \Big)$$

$$= \sqrt{(-1-1)^2 + (0-1)^2 + (1-(-1))^2 + (0-0)^2}$$

$$= \sqrt{(-2)^2 + 1^2 + 2^2 + 0^2}$$

$$= \sqrt{9} = 3$$

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

where sb is string concatenation.

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

function name DOMAIN

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

Rule

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

function application value

The domain of *rnalen* is S

The codomain of *rnalen* is $\mathbb{Z}^+$

Example function application:

$$rnalen(\texttt{ACU}) = 1 + rnalen(AC)$$

$s = AC$ b recursive step of definition of rnalen

$s = A$ b $\in B$

$$= 1 + (1 + rnalen(A))$$ recursive step of definition of rnalen

$$= 1 + (1 + 1)$$ using the basis step in definition of rnalen because $A \in B$

$$= 3$$

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

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

$\text{basecount}((\underbrace{A}_{s}\underbrace{C}_{}\underset{b_1}{U}, \underset{b_2}{C}))$

$= \text{basecount}((\underset{s}{\underline{A}}\underset{b_1}{C}, \underset{b_2}{C}))$ by the second option in the recursive step, with $s = AC$, $b_1 = U$, $b_2 = C$

$= 1 + \text{basecount}((\underset{b_1}{A}, \underset{b_2}{C}))$ by the first option in the recursive step, with $s = A$, $b_1 = C$, $b_2 = C$

$= 1 + 0$ by the second option in the basis step, with $b_1 = A$ and $b_2 = C$

$= 1$