This page has some useful notation that will be used throughout the course. Find the definitions for each of these terms by looking in the index of the course textbook.

Term	Notation Example(s)	We say in English
n-tuple	(x_1, x_2, x_3)	The 3-tuple of x_1, x_2, and x_3
	$(3, 4)$	The 2-tuple or **ordered pair** of 3 and 4
sequence	$x_1, \ldots, x_n$	A sequence x_1 to x_n
	$x_1, \ldots, x_n$ where $n = 0$	An empty sequence
	$x_1, \ldots, x_n$ where $n = 1$	A sequence containing just x_1
	$x_1, \ldots, x_n$ where $n = 2$	A sequence containing just x_1 and x_2 in order
	x_1, x_2	A sequence containing just x_1 and x_2 in order
set		Unordered collection of objects. The set of ...
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.
roster method	$\{43, 7, 9\}$	The set whose elements are 43, 7, and 9
	$\{9, \mathbb{N}\}$	The set whose elements are 9 and $\mathbb{N}$
set builder notation	$\{x \in \mathbb{Z} \mid x > 0\}$	The set of all x from the integers such that x is greater than 0
	$\{3x \mid x \in \mathbb{Z}\}$	The set of all integer multiples of 3 **Note**: we use the convention that writing two numbers next to each other means multiplication.
function definition	$f(x) = x + 4$	Define f of x to be $x + 4$
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
summation notation	$\sum_{i=1}^{n} i$	The sum of the integers from 1 to n, inclusive
	$\sum_{i=1}^{n} i^2 - 1$	The sum of $i^2 - 1$ (i squared minus 1) for each i from 1 to n, inclusive
quotient, integer division	n **div** m	The (integer) quotient upon dividing n by m; informally: divide and then drop the fractional part
modulo, remainder	n **mod** m	The remainder upon dividing n by m

Themes for CSE 20

- Technical skepticism
- Multiple representations

Recurring examples in CSE 20

- Clustering and recommendation systems (machine learning, Netflix)
- Genomics and bioinformatics (DNA and RNA)
- Codes and information (secret message sharing and error correction)
- "Under the hood" of computers (circuits, pixel color representation, data structures)

This week's highlights

- Use and apply definitions and notation
- Explore mathematical definitions related to a specific application (Netflix)
- Define data types: set, n-tuple, string (over specific alphabet)
- Define sets and functions in multiple ways
- Trace an algorithm specified in pseudocode
- Define the base expansion of a positive integer, specifically decimal, binary, hexadecimal, and octal.
- Convert between expansions in different bases of a positive integer.
- Define and use the div and mod operators.

Lecture videos

Week 1 Day 1 YouTube playlist[1]

Week 1 Day 2 YouTube playlist[2]

Week 1 Day 3 YouTube playlist[3]

[1] https://youtube.com/playlist?list=PLML4QilACLk7gzYrukE78l8mPniy1ieP2
[2] https://youtube.com/playlist?list=PLML4QilACLk5UEuC7vC3KVh4-c-4omZN8
[3] https://youtube.com/playlist?list=PLML4QilACLk6J5h3Jg-m71pMXWKvPwXDI

Monday January 4

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).

Person	Fyre	Frozen II	Picard	Ratings written as a 3-tuple
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.

Define the following functions whose inputs are ordered pairs of 3-tuples each of whose components comes from the set $\{-1, 0, 1\}$

$$d_1((x_1, x_2, x_3), (y_1, y_2, y_3)) = \sum_{i=1}^{3} ((|\ x_i - y_i\ | + 1)\ \textbf{div}\ 2)$$

sum, absolute value

$$d_2(\ (x_1, x_2, x_3), (y_1, y_2, y_3)\ ) = \sqrt{\sum_{i=1}^{3} (x_i - y_i)^2}$$

$d_1(P_4, P_1)$ $x_1=0\ x_2=-1\ x_3=1$ $y_1=-1\ y_2=0\ y_3=1$ $(1+1)\text{div}2 + (1+1)\text{div}2 + (0+1)\text{div}2$ $= 1+1+0 = 2$	$d_1(P_4, P_2)$	$d_1(P_4, P_3)$
$d_2(P_4, P_1)$	$d_2(P_4, P_2)$	$d_2(P_4, P_3)$ $x_1=0\ x_2=-1\ x_3=1$ $y_1=1\ y_2=1\ y_3=1$ $\sqrt{(-1)^2 + (-2)^2 + 0^2} = \sqrt{5}$

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 the new functions that would be used to compare the 4-tuples of ratings encoding movie preferences now that there are four movies in the database.

Wednesday January 6

Term	Examples: (add additional examples from class)
set unordered collection of elements *Equal means agree on membership of all elements*	$7 \in \{43, 7, 9\}$ $\quad 2 \notin \{43, 7, 9\}$ $\{1, 85, 132\} = \{85, 1, 132\} = \{1, 85, 132, 1\}$ $\emptyset$
n-tuple ordered sequence of elements with n "slots" *Equal means corresponding components equal*	$(1, 12)$ 2-tuple aka ordered pair $(1, 12, 1)$ 3-tuple different (unequal)
string ordered finite sequence of elements each from specified set *Equal means same length and corresponding characters equal*	CSE20

$\{-1, 1\}$ $\quad \{0, 0\}$ $\quad \{-1, 0, 1\}$ $\quad \mathbb{Z}$ $\quad \mathbb{N} = \{x \in \mathbb{Z} \mid x \geq 0\}$ $\quad \emptyset$ $\quad \mathbb{Z}^+ = \{x \in \mathbb{Z} \mid x > 0\}$

Which of the sets above are defined using the roster method? Which are defined using set builder notation?

Which of the sets above have 0 as an element?

Can you write any of the sets above more simply? $\{0, 0\} = \{0\}$

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

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: AU $\quad$ A

To define a set we can use the **roster method**, the **set builder notation**, and also ...
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 a rule 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.

Extra example: The set of binary strings, denoted $\{0,1\}^*$ which we read as "zero one star", is defined (recursively) by:

Basis Step: $\lambda \in \{0,1\}^*$
Recursive Step: If $s \in \{0,1\}^*$ then $s0 \in \{0,1\}^*$ and $s1 \in \{0,1\}^*$

where $s0$ and $s1$ are the results of string concatenation. The symbol λ, pronounced "lambda" is used to denote the empty string and has the property that $\lambda x = x\lambda = x$ for each string x.

Examples: the following are each binary strings

001 λ 11

To define a set we can use the roster method, the 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 (Rosen p. 123) Let A and B be sets. The **Cartesian product** of A and B, denoted $A \times B$, is the set of all ordered pairs (a, b) where $a \in A$ and $b \in B$

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

each element of $A \times B$ is an ordered pair

Definition: Let A and B be sets of strings over the same alphabet. The **set-wise concatenation** of A and B, denoted $A \circ B$, is the set of all results of string concatenation ab where $a \in A$ and $b \in B$

$$A \circ B = \{ab \mid a \in A \text{ and } b \in B\}$$

setwise concatenation

each element of $A \circ B$ is a string

Set	Example elements in this set:			
B	A	C	G	U
$\{(A,C),(U,U)\}$	(A, C)		(U, U)	
$B \times \{-1,0,1\}$		$(U,-1)$	$(C,0)$	
$\{-1,0,1\} \times B$			$(0,C)$	
			$(0,0,0)$	
$\{A,C,G,U\} \circ \{A,C,G,U\}$	AA		CU	
$\{G\} \circ \{G\} \circ \{G\} \circ \{G\}$ ALT	GGGG			
$\{A\} \circ \{U\}$	AU			

$(0,C)$ NOT AN ELEMENT

12 elements in Cartesian product

UA NOT AN ELEMENT

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.

inputs — possible outputs

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

Examples:

input → function → output

Define the function

f: domain → codomain

as <<fill in rule>>

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

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

s = AC, b = U

s = A, b = C

$= 1 + (1 + rnalen(\texttt{A}))$ rec step

$= 1 + (1 + 1) = 3$

basis step.

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

$basecount : S \times B \to$

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

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

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

positional representation
b = 2 or 10 or ...

Friday January 8

base

Definition (Rosen p. 246) 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

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

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

Binary $b = 2$	Octal $b = 8$	Decimal $b = 10$	Hexadecimal $b = 16$
$(1401)_2$ 4 is not allowed coefficient for b2			
$1 \cdot 512 + 1 \cdot 256 + 1$ $= 2^9 + 2^8 + 2^0$ $= (110000000001)_2$	$(1401)_8$	$1 \cdot 8^3 + 4 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0$ $= 512 + 256 + 0 + 1$ $= (769)_{10}$ thousands hundreds tens ones	$3 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0$ $= (301)_{16}$
$1401 = 1024 + 256 + 64 + 32 + \cdots$ $= 2^{10} + 2^8 + 2^6 + 2^5 + 2^4 + 2^3 + 2^0$ $= (10101111001)_2$ $2^{12} = 4096$ $2^{10} = 1024$ $2^0 = 1$ $5121 = 2^{12} + 2^{10} + 2^0$ $= (1010000000001)_2$	$1401 = 2 \cdot 512 + 5 \cdot 8^2 + 7 \cdot 8 + 1$ $= 2 \cdot 8^3 + 5 \cdot 8^2 + 7 \cdot 8^1 + 1 \cdot 8^0$ $= (2571)_8$ $8^3 = 512$ $8^4 = 4096$ $5121 = 4096 + 2 \cdot 512 + 1$ $= 1 \cdot 8^4 + 2 \cdot 8^3 + 1 \cdot 8^0$ $= (12001)_8$	$(1401)_{10}$ 10^3 10^2 10^1 10^0 1401	$1401 = 5 \cdot 16^2 + 7 \cdot 16 + 9$ $= (579)_{16}$ $1 \cdot 16^3 + 4 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0$ $= 4096 + 4(256) + 1$ $= 4096 + 1024 + 1$ $= (5121)_{10}$

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

Powers of 2: $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 32, 2^6 = 64, \ldots 2^{10} = 1024$

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 := n div 2
6  return r {r holds the result of the log operation}
```

n	r	$n > 1$?
6	0	6 > 1 ? T
3	1	3 > 1 ? T
1	2	1 > 1 ? F
return	2	

Note: $2^2 < 6 < 2^3$ so $\log_2 6 = 2.\ldots$ so int part is 2

Two algorithms for constructing base b expansion from decimal representation

Algorithm 1: Start with highest power of b, i.e. at 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 > 1
4     r := r + 1
5     n := n div b
```

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}
```

The Division Algorithm (Rosen 4.1 Theorem 2, p. 239) 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?

Algorithm 2: Start with right-most coefficient of expansion

n	b	q	k	a_k	$q \neq 0$?
17	3	17	0	$a_0 = 17 \bmod 3 = 2$	T
		$17 \text{ div } 3 = 5$	1	$a_1 = 5 \bmod 3 = 2$	T
		$1 \text{ div } 3 = 0$	2	$a_2 = 1 \bmod 3 = 1$	F

$17 = (122)_3$

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}
```

Idea: (when $k > 1$) $n = a_{k-1}b^{k-1} + \cdot + a_1 b + a_0 = b(a_{k-1}b^{k-2} + \cdots + a_1) + a_0$ so $a_0 = n$ **mod** b and $a_{k-1}b^{k-2} + \cdots + a_1 = n$ **div** b.

the base 3 expansion of

Using Algorithm 1 to calculate $(17)_{10}$

$V = 17\ 17\ 5\ 2\ 0$

$k = \log_3(17.3) + 1 = 2 + 1 = 3$

$i = 1$: $a_2 = 1$; $17 \geq 3^2$? T; $8 \geq 3^2$? F

$i = 2$: $a_1 = 2$; $8 \geq 3^1$? T; $5 \geq 3^1$? T; $2 \geq 3^1$? F

$i = 3$: $a_0 = 2$; $2 \geq 3^0$? T; $1 \geq 3^0$? T; $0 \geq 3^0$? F

the base 3 expansion of

Using Algorithm 2 to calculate $(17)_{10}$

Review quiz questions

1. Please complete the beginning of the quarter survey https://forms.gle/8qnCRC26muASsXFi8.

2. Consider $n = 5$. Define the following functions whose inputs are ordered pairs of 5-tuples each of whose components comes from the set $\{-1, 0, 1\}$

$$d_{1,5}((x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)) = \sum_{i=1}^{5} ((|\ x_i - y_i\ | + 1)\ \textbf{div}\ 2)$$

$$d_{2,5}(\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ ) = \sqrt{\sum_{i=1}^{5} (x_i - y_i)^2}$$

   (a) Consider the function application $d_{1,5}(\ (1, 1, 1, 1, 1), (1, 1, 1, 1, 1)\ )$
      i. What is the input?
      ii. What is the output?

   (b) Consider the function application $d_{2,5}(\ (1, 0, 1, 1, 1), (0, 0, 1, 1, 1)\ )$
      i. What is the input?
      ii. What is the output?

   (c) What is the domain of the function $d_{1,5}$?
      i. $\{-1, 0, 1\}$
      ii. $\{-1, 0, 1\} \times \{-1, 0, 1\}$
      iii. $\{-1, 0, 1\} \times \{-1, 0, 1\} \times \{-1, 0, 1\} \times \{-1, 0, 1\} \times \{-1, 0, 1\}$
      iv. None of the above.

   (d) **True** or **False**: The functions $d_{1,5}$ and $d_{2,5}$ have the same domain.

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

   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 $sb \in S$

   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.

4. For many applications in cryptography and random number generation, dividing very large integers efficiently is critical. Recall **The Division Algorithm** (Rosen 4.1 Theorem 2, p. 239): 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 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 n is any positive integer 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.*

5. Colors can be described as amounts of red, green, and blue mixed together[4]. 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:

[4]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`

{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

$$\{y \in \mathbb{Z} \mid -1 < y \leq 255\}$$

.