WORKSHEET WEEK 1 MONDAY
CSC165 — 2025 WINTER

Developing a program requires problem-solving and understanding the programming language in which you are expressing your solution. This is also the case for developing a proof, which is a precise description of the solution to a precise problem, using mathematical language.
You are aware that there is no mechanical/algorithmic/formulaic procedure/recipe for you to develop a program. This is also the case for developing a proof.

Let's dive into some problem-solving, while paying attention to the role of parameters in mathematical language.

Problem-Solving

Here's a problem:

> Let n be a natural number. For which matrices
> $$a = \begin{pmatrix} a_{0,0} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{n-1,0} & \cdots & a_{n-1,n-1} \end{pmatrix}$$
> containing natural numbers, is
> $$\sum_{j=0}^{n-1} \left(\prod_{k=0}^{n-1} a_{j,k} \right) = 0\,?$$

Note that in our course the set of **natural numbers** $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$, which includes zero.
And recall that, for start and end integers s and e with $s \leq e + 1$, the **summation notation** $\sum_{i=s}^{e} t_i$ is an abbreviation of $t_s + t_{s+1} + \cdots + t_e$, and the **product notation** $\prod_{i=s}^{e} t_i$ is an abbreviation of $t_s \cdot t_{s+1} \cdot \cdots \cdot t_e$.
We'll refer to the unabbreviated expressions as the **expansions** of the notations.

First, we identify which variables in the problem are **parameters** of the problem.
The first variable mentioned is n, without a specific value but with the **precondition** that it's a natural number. It appears to have the features you're familiar with for parameters of functions in programming. To see if it behaves as a parameter let's try the mathematical analogue of function call: **instantiate** the problem with a value (that satisfies the precondition) for the variable, which means picking a specific natural number and **syntactically** (textually) substituting (replacing) the variable name with the value in the body (the part of the problem without the **declaration** of the parameter).
Do that now: instantiate the problem with $n = 165$, by: copying the problem text, omitting the "Let n be a natural number.", and replacing each occurrence of "n" with "165"; if you're doing this on paper and don't want to write much just annotate the problem above by writing "165" above/beside/underneath each "n".

> Instance $n = 165$ of the problem.
>
> Let $\overset{165}{n}$ be a natural number. For which matrices $a = \begin{pmatrix} a_{0,0} & \cdots & a_{0,\overset{165}{n}-1} \\ \vdots & \ddots & \vdots \\ a_{\overset{165}{n}-1,0} & \cdots & a_{\overset{165}{n}-1,\overset{165}{n}-1} \end{pmatrix}$
> containing natural numbers, is $\sum_{j=0}^{\overset{165}{n}-1} \left(\prod_{k=0}^{\overset{165}{n}-1} a_{j,k} \right) = 0\,?$
>
> The key is that the variable literally stands for some number, it could go away completely:
>
> For which matrices $a = \begin{pmatrix} a_{0,0} & \cdots & a_{0,165-1} \\ \vdots & \ddots & \vdots \\ a_{165-1,0} & \cdots & a_{165-1,165-1} \end{pmatrix}$ containing natural numbers,
> is $\sum_{j=0}^{165-1} \left(\prod_{k=0}^{165-1} a_{j,k} \right) = 0\,?$

Since we have a specific number we can calculate the values of the "$n-1$" expressions.
Do that now: replace each "$165-1$" with "164".

Instance $n=165$ of the problem.
For which matrices $a = \begin{pmatrix} a_{0,0} & \cdots & a_{0,164} \\ \vdots & \ddots & \vdots \\ a_{164,0} & \cdots & a_{164,164} \end{pmatrix}$ containing natural numbers,
is $\sum_{j=0}^{164}\left(\prod_{k=0}^{164} a_{j,k}\right) = 0\,$?

We now have the variable a without a specific value but with a specific precondition, so it looks like a parameter as well. There are various ways we could describe a specific a, e.g., "every element of a is 123" or "$a_{j,k} = j+k$ for each $j,k \in \{0,\ldots,164\}$", or we could fill in some elements and leave some "$\cdots$"s to suggest some pattern. But let's back up and perform a standard problem-solving technique: instantiate the problem with small/simple values that are hopefully **generic** enough to be **representative** of the general situation. Let's try $n=3$:

Instance $n=3$ of the problem.
For which matrices $a = \begin{pmatrix} a_{0,0} & \cdots & a_{0,2} \\ \vdots & \ddots & \vdots \\ a_{2,0} & \cdots & a_{2,2} \end{pmatrix}$ containing natural numbers,
is $\sum_{j=0}^{2}\left(\prod_{k=0}^{2} a_{j,k}\right) = 0\,$?

That value is small enough that we can write out all the elements implicit in the uses of "$\cdots$" notation, and expand the sum and product notations. Do that now.

Instance $n=3$ of the problem.
For which matrices $a = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{pmatrix}$ containing natural numbers, is

$$\sum_{j=0}^{2}\left(\prod_{k=0}^{2} a_{j,k}\right), \text{ i.e., } \prod_{k=0}^{2} a_{0,k} + \prod_{k=0}^{2} a_{1,k} + \prod_{k=0}^{2} a_{2,k}\ , \text{ i.e., } \begin{pmatrix} & a_{0,0}\cdot a_{0,1}\cdot a_{0,2} \\ + & a_{1,0}\cdot a_{1,1}\cdot a_{1,2} \\ + & a_{2,0}\cdot a_{2,1}\cdot a_{2,2} \end{pmatrix} = 0\,?$$

Notice we expanded the Σ and then the Πs. If you did it as one step that's fine, but in more complicated situations you might find it easier and more reliable to do it in steps. And, depending on the situation, and how comfortable you are with the unexpanded notation (possibly imagining it expanded on your mental whiteboard), you might choose to only expand some parts.

Instantiate a with natural number elements of your choosing, but try to avoid making it too special.

Instance $n=3$, $a = \begin{pmatrix} 1 & 6 & 5 \\ 8 & 0 & 1 \\ 4 & 1 & 8 \end{pmatrix}$. Is $\begin{pmatrix} 1\cdot 6\cdot 5 \\ +8\cdot 0\cdot 1 \\ +4\cdot 1\cdot 8 \end{pmatrix} = 0$?

Notice we didn't simplify the sum of products, choosing to **delay their evaluation**. Don't automatically perform calculations or "simplifying" algebraic manipulations, unless you determine they're worthwhile. This can not only save effort, but often preserves useful structural information.

Determine whether the equation is true for your example, and for the one in our solution. Give an explanation of why, that would help *someone else* see why as easily as possible.

> $\begin{pmatrix} 1\cdot 6\cdot 5 \\ +8\cdot 0\cdot 1 \\ +4\cdot 1\cdot 8 \end{pmatrix} \neq 0$ since it's the sum of a **positive** number (from a product of positive numbers), zero (from a product including a zero), and a positive number (from a product of positive numbers), so is positive.

Inspired by the ideas in our description, give an example matrix (still with $n = 3$) for which the equation is false, trying to have as few positive elements as possible; and give an example for which it's false, with as few zero elements as possible.

$$\begin{pmatrix} 1\cdot 6\cdot 5 \\ +0\cdot 0\cdot 0 \\ +0\cdot 0\cdot 0 \end{pmatrix} \neq 0, \begin{pmatrix} 0\cdot 6\cdot 5 \\ +8\cdot 0\cdot 1 \\ +4\cdot 0\cdot 8 \end{pmatrix} = 0$$

Try to find simple descriptions of the matrices for which the equation is true, and for which the equation is false, referring to the rows and/or columns but without referring to sums or products. Make it easy for a careful ten-year-old to use to determine whether the equation is true or not, without that ten-year-old needing to know the equation or even what addition and multiplication are.

> It's true when each row has at least one zero, false when at least one row has no zeros.
> As programmers you express solutions in terms of a particular language with a particular set of libraries, and we can think of the libraries as a dictionary of terminology. Analogously, we asked for the solution in the English of ten-year-olds, in terms of rows and/or columns. Solving a problem is about transforming it into a form that has certain advantages (e.g., into an algorithm to be programmed in a certain language with certain libraries, or a description a ten-year-old can understand.
> Soon in the course we'll cover a **symbolic** language that has notations for concepts referred to by the "each", "at least one", and "no" in our **prose** description.

Coming up with **good examples** when problem-solving is similar to coming up with **good test cases** in Software Engineering. Although the "test" in "test cases" emphasizes their use for checking a solution, Test-Driven Development recognizes their value for understanding a problem and developing a (programmatic) solution as well. We'll often use the phrase "good set of examples" and "good set of test cases" interchangeably in this course, regardless of whether we're referring specifically to a programming problem.
Tests cases are often classifed as **edge/boundary/special/extreme/corner** versus generic/representative. Notice that the last pair of examples in our case were on "edges" or "boundaries": changing one non-zero element to zero in the first example changes the sum to zero, and changing one zero element to non-zero in the second example changes the sum to non-zero. They were asked for as extremes ("as few positive / zero elements"). It turned out that the positions of the zeros matters more than just the total number of them, but, until we determined that, trying to minimize or maximize the total number seemed a natural first step to finding the boundaries of the problem.

What would you consider the edge cases for the parameter n?

> $n = 1$ involves no products nor additions; the other side of that is $n = 2$ which turns out to be a good instance for a minimal generic exploration.
> $n = 0$ is quite different from all the others: because the sum and product then have *conventions* for their interpretation, and we're not sure that our reasoning, summarized as looking at zeros in rows, is correct or even sensible.

Take your or our summary of how to determine whether the equation is true or false, and interpret it for matrices that have no elements: do the summaries have a meaning, and if so what do they claim?

> "when at least one row contains no zeros" seems to make sense, and be false; that was our condition for the equation to be false, so it would say the equation is true.
> "when each row contains at least one zero" is perhaps less clear when there are no rows at all, but if we look at it as an obligation (check each row, make sure it doesn't fail) then it appears to be true. So that's consistent.

Recall that for an integer s, $\sum_{i=s}^{s-1} t_i$ is defined to be zero *by convention*, and $\prod_{i=s}^{s-1} t_i$ is defined to be one. Instantiate the equation with $n = 0$ and use the conventions to determine its value. Does this the match the summaries?

> For $n = 0$ the LHS of the equation is $\sum_{j=0}^{-1}\left(\prod_{k=0}^{-1} a_{j,k}\right)$, which is an instance of the convention for $\sum$ (with $s = 0$, index variable j, and term expression $t_j = \prod_{k=0}^{-1} a_{j,k}$), so is zero.
> So that convention makes the equation true, which happens to match our summaries. If it didn't match our summaries we would add a case to them (for $n = 0$ the equation is true, for $n \geq 1$ it's true when at least one row ..."; "the equation is false when $n \geq 1$ and each row ...").
> If you're unsure about the intepretation and result of the summaries for $n = 1$, since it can seem quite special, and whether the actually class the matrices according to the truth of the equation, check that.

Homework.

> *NOTE: In these worksheets, we will use the word "homework" NOT to mean something you do to earn grades, but something you do to improve your understanding of the course material.*

Expand only the Π notation in $\sum_{j=0}^{2}\left(\prod_{k=0}^{2} a_{j,k}\right)$.

Expand $\sum_{j=0}^{n-1}\left(\prod_{k=0}^{n-1} a_{j,k}\right)$, using "$\cdots$" notation since you don't know the specific number of terms.

Explore $\sum_{j=0}^{n-1}\left(\sum_{k=0}^{n-1} a_{j,k}\right) = 0$, $\prod_{j=0}^{n-1}\left(\sum_{k=0}^{n-1} a_{j,k}\right) = 0$, and $\prod_{j=0}^{n-1}\left(\prod_{k=0}^{n-1} a_{j,k}\right) = 0$; do as much instantiation and expansion as you need to understand them, and for each one give a good pair of example matrices at the edges of true and false along with summaries of the matrices for which they're true and for which they're false. Be sure to check the edge case $n = 0$ for each one, and add a case to any of your summaries whose meaning is unclear or incorrect for that case.

Let's call the original equation $E_{\Sigma\Pi}$, and these three equations $E_{\Sigma\Sigma}$, $E_{\Pi\Sigma}$, and $E_{\Pi\Pi}$.

Do any of the equations **entail** another one of them? E.g., for every instance for which $E_{\Sigma\Pi}$ is true is that instance of $E_{\Sigma\Sigma}$ (the one from the same setting of the parameters) also true? For the ones that don't, give a **counter-example**, e.g., if $E_{\Sigma\Pi}$ doesn't entail $E_{\Sigma\Sigma}$ give a specific instance where $E_{\Sigma\Pi}$ is true but $E_{\Sigma\Sigma}$ is false. That's twelve comparisons, but some of the equations might be easy to compare to multiple other equations at once.

Definitions

Definitions are also parameterized, and their meanings are also derived via instantiation.
Let's start by recalling **set comprehension** (aka set-builder) notation and **set enumeration** notation.
Instantiate the following expression with $x = 1.65$: $\{z \in \mathbb{Z} : z \leq x\}$.

> $\{z \in \mathbb{Z} : z \leq 1.65\}$

The variable $\mathbb{Z}$ isn't a parameter, it already has a value in mathematics as the set of Integers. Our "Numeric Types" reference gives the definition of $\mathbb{Z}$'s value using set enumeration notation, as $\{0, -1, 1, -2, 2, -3, 3, \ldots\}$.
Instantiate your previous expression with that (expression for the) value of $\mathbb{Z}$, and then **expand** the result as a set enumeration.

> $\{z \in \{0, -1, 1, -2, 2, -3, 3, \ldots\} : z \leq 1.65\} = \{0, -1, 1, -2, -3, \ldots\} = \{1, 0, -1, -2, -3, \ldots\}$.
> We demonstrated a step where we took the elements as originally enumerated and simply included only the elements that were at most 1.65, to emphasize the general operation of a set comprehension. But the rearrangement is clearer. When using "…", which elements to omit, and which to list and in what order, requires judgement.

Consider the following definitions:

$$\text{For } x \in \mathbb{R},\ \lfloor x \rfloor = \max\{z \in \mathbb{Z} : z \leq x\},$$
$$\text{and } \lceil x \rceil = \min\{z \in \mathbb{Z} : x \leq z\}.$$

Instantiate those definitions with $x = 1.65$ and the definition of $\mathbb{Z}$, expand the set comprehensions, then determine the final value by taking the maximum and minimum of the respective sets.

> $\lfloor x \rfloor = \max\{z \in \{0, -1, 1, -2, 2, -3, 3, \ldots\} : z \leq 1.65\} = \max\{1, 0, -1, -2, -3, \ldots\} = 1$.
> $\lceil x \rceil = \min\{z \in \{0, -1, 1, -2, 2, -3, 3, \ldots\} : 1.65 \leq z\} = \min\{2, 3, 4, \ldots\} = 2$.

Work out $\lfloor x \rfloor$ and $\lceil x \rceil$ for as any examples as you need to understand them. You may skip any parts of the instantiations and expansions that *you* don't find necessary.

> This varies from student to student, but since the definitions involve integers at least one good edge case would be an integer, e.g., $\lfloor 165 \rfloor = 165 = \lceil 165 \rceil$.
> Another good test case is a negative number: $\lfloor -1.65 \rfloor = -2$ and $\lceil -1.65 \rceil = -1$ to emphasize that floor doesn't just remove the fractional part from the number's *decimal representation.*
> These functions are called the **floor** and **ceiling**, respectively: the floor of x is the largest integer which is at most x, and the ceiling is the smallest integer that is at least x.

Homework.
Familiarize yourself with the sets defined at the start of our "Numeric Types" reference, and make sure you understand all the uses of notation.
You might not have encountered "$\setminus$" before, which is **set difference**: for sets A and B, $A \setminus B$ is defined as $\{a \in A : a \notin B\}$. Expand $\mathbb{R} \setminus \{0\}$ accordingly, and try to simplify the resulting set comprehension slightly.
To get comfortable with set difference you could make a good set of test cases for it (what's an edge case for a set, what's an edge case for a pair of sets, and so on).
To help remember the meanings of the superscripted versions of the four named sets, you could try to express each of those versions ($4 \times 3 = 12$ in total) in some way (besides the prose descriptions we gave). Are any of the sets redundant (the same as one of the others or one of the originals)? Some of them are already shown as enumerations or with interval notations: you might consider for each one whether you can give a nice enumeration, set comprehension, and/or some other expression.

For the three examples for floor and ceiling, the number and its floor and ceiling were either all the same, or the floor was one less than the ceiling and the number was in-between. Is that always true? If not, give a counter-example.

Let $x \in \mathbb{R}$. Give bounds on $\lfloor x \rfloor$ in terms of x, by filling in the blanks in the following with expressions involving x (without using $\lfloor x \rfloor$ or $\lceil x \rceil$): $\_\_ \leq \lfloor x \rfloor \leq \_\_$. Can either of those be strengthened to be **strict** inequalities? Give bounds on x in terms of $\lfloor x \rfloor$.

If $x \in \mathbb{R}$, is $\lfloor -x \rfloor = -\lfloor x \rfloor$ (always)? If not, give a counter-example, and see if you can find some relatively simple formula for $\lfloor -x \rfloor$.

If $x, y \in \mathbb{R}$, is $\lfloor x + y \rfloor = \lfloor x \rfloor + \lfloor y \rfloor$? If not, give a counter-example, and see if you can find a *simple* restriction on y to make it true (start by instantiating the problem with some good examples, what are some edge cases for $\lfloor x \rfloor$, $\lfloor y \rfloor$, $\lfloor x + y \rfloor$, etc.). Is that restriction on y necessary (is there an x and a y without that restriction where it's true)?

If $x, y \in \mathbb{R}$, is $x + y = \lfloor x \rfloor + \lceil y \rceil$? If not, give a counter-example, and see if you can find a simple restriction on x, y, and/or $x + y$ to make it true. Is that restriction necessary?

Propositional Logic: Conjunction and Disjunction

In this section of the worksheet, let $a, b \in \mathbb{N}$.
In particular, a and b are at least zero (**non-negative**), i.e., $a \geq 0$ and $b \geq 0$.
If we want to express a condition that a is positive (*while knowing that* $a \in \mathbb{N}$) there are three natural inequalities we can write; find and state those inequalities symbolically and in natural prose.

$a > 0$	a is positive ; a is greater than zero
$a \geq 1$	a is at least one ; a is greater than or equal to one
$a \neq 0$	a is non-zero ; a isn't zero ; a is unequal to zero ; a is not equal to zero

Notice that for the prose we wrote grammatically correct phrases — we didn't just mechanically write out the symbols in English. E.g., we didn't just write "a greater than zero" (notice the lack of "is").
Symbolic notation is just *precise shorthand*; don't let its form distract you from processing its meaning. Although most notation can be expressed in prose by a fairly mechanical left-to-right symbol-by-symbol translation (see our final prose alternative for each of the three conditions) that doesn't necessarily produce the most natural or even the shortest prose.

Find two inequalities equivalent to the condition $a = 0$ (still assuming that $a \in \mathbb{N}$), and state them symbolically and in prose.

$a < 1$	a is less than one
$a \leq 0$	a is at most zero ; a is less than or equal to zero ; a is no more than zero

If $a > 0$ and $b > 0$ what's the **strongest** claim you can make about $a + b$?

$a + b$ is a natural number and $a + b \geq 2$, since the smallest that we can make the sum is when both of a and b are 1.
Although it's true that $a + b > 0 + 0 = 0$, which is equivalent to $a + b \geq 1$ since the sum is a natural number, each of a and b is at least 1 so the sum can't be exactly 1. So we can claim that $a + b > 1$, or equivalently here $a + b \geq 2$, which is stronger than just $a + b > 0$ or $a + b \geq 1$.
If we had started by characterizing the conditions on a and b as $a \geq 1$ and $b \geq 1$ we could deduce that $a + b \geq 1 + 1 = 2$, and it's actually possible that $a + b = 2$ (when $a = b = 1$) so we can't strengthen that claim any more (except, perhaps, to point out also that $a + b$ is a natural number).

If $a < 10$ and $b < 10$, what's the strongest claim you can make about $a + b$?

$a + b \in \mathbb{N}$ and $a + b \leq 18$ (or with **strict** inequality: $a + b < 19$)

In prose, to combine two conditions into one condition that expresses that both conditions are true, we can use "and". Combining two conditions this way is called **conjunction** and the two conditions are **conjuncts**. Symbolically, we use the binary conjunction operator "$\wedge$" (read as "and"); so, e.g., "$a \geq 0$ and $b \geq 0$" can be expressed symbolically as $a \geq 0 \wedge b \geq 0$.

Conjunction combines *only* true/false (boolean) expressions (**propositions**), which is why we don't need parenetheses to disambiguate $a \geq 0 \wedge b \geq 0$: none of $a \geq (0 \wedge b) \geq 0$, $a \geq (0 \wedge (b \geq 0))$, nor $((a \geq 0) \wedge b) \geq 0$ make sense since they each have a *number* as a conjunct (and even if, e.g., $0 \wedge b$ was valid its result would be a boolean and then $a \geq (0 \wedge b)$ would be an invalid inequality). Similarly, trying to express "a and b are at least zero" as $a \wedge b \geq 0$ is incorrect: explain why.

The two potential interpretations are $a \wedge (b \geq 0)$, which tries to conjoin a number, and $(a \wedge b) \geq 0$, which tries to conjoin two numbers (and tries to compare the result of a conjunction, which if valid would be a boolean, to a number).

Fully-parenthesize (put parentheses in, to make the **sub-expressions** fully explicit) the only valid interpretation of $a \geq 0 \wedge b \geq 0$, and state the **datatype** of each of its sub-expressions (including the whole expresssion).

> $(a \geq 0) \wedge (b \geq 0)$; a, 0, and b are (natural) numbers ; $(a \geq 0)$, $(b \geq 0)$, and $(a \geq 0) \wedge (b \geq 0)$ are boolean. We can annotate the structure of the expression with the datatypes of the sub-expressions as follows ($\mathbb{B}$ is the set of booleans $\{\textbf{true}, \textbf{false}\}$):
>
> $$\overset{\mathbb{B}}{\overline{\overset{\mathbb{B}}{\overline{\overset{\mathbb{N}}{\overline{a}} \geq \overset{\mathbb{N}}{\overline{0}}}} \wedge \overset{\mathbb{B}}{\overline{\overset{\mathbb{N}}{\overline{b}} \geq \overset{\mathbb{N}}{\overline{0}}}}}}$$

Earlier we wrote "$a, b \in \mathbb{N}$": express that as a conjunction, both fully-parenthesized and also with the minimal necessary parenthesization, and state the datatype of each of its (sub-)expressions.

> $(a \in \mathbb{N}) \wedge (b \in \mathbb{N})$; $a \in \mathbb{N} \wedge b \in \mathbb{N}$; a and b are numbers ; $\mathbb{N}$ is a set (of numbers) ; $(a \in \mathbb{N})$, $(b \in \mathbb{N})$, and the whole expression, are boolean.
>
> *For any expression you read or write, make sure you can identify its sub-expressions (and be able to full-parenthesize the expression to make them explicit) and their datatypes. When writing an expression use as much parenthesization as you like to make the structure of the expression explicit, and in expressions you read (including ones we give you on tests) add parentheses if you'd like to make some implicit parenthesization explicit.*

How would you describe the values of a and b for which $a + b \geq 1$, if the only inequalities you can use compare a with a specific number or b with a specific number (but don't compare a and b to each other nor involve arithmetic formulas that combine a and b)?

> One approach: "a is positive and/or b is positive".
> As noted earlier, since a and b are natural numbers there are some valid alternatives to "positive". When we say "valid" we don't mean "would receive full marks in this course if given as an answer to this question" (although they would), we mean "communicates correctly to other human beings" — which is what we want to learn.
> There are also many alternative prose phrasings that have the same logical structure, e.g., "one or both of a and b are positive", "at least one of a and b is positive".
> Another appoach, which has a different logical structure: "a and b are not both zero". We'll consider this approach in more detail in Worksheet 1.

To express that *at least one* (*one or both*) of two conditions is true we use **disjunction**, written symbolically as "$\vee$", and then the conditions are called **disjuncts**. We can describe the values of a and b for which $a + b \geq 1$ as the values for which $a \geq 1 \vee b \geq 1$. This has the same meaning as Python's `or`, in particular it's **inclusive**. In English the meaning of "or" is very context-dependent, especially whether it's inclusive or **exclusive**, but if you intend the inclusive meaning you can use "and/or" to make that explicit.

Recall equation $E_{\Sigma\Pi}$ from the PROBLEM-SOLVING section. Use conjunction and/or disjunction to express, symbolically, when instance $n = 3$ is true, in terms of conditions on individual elements of a. Repeat this for when it's false.

$$(a_{0,0} = 0 \vee a_{0,1} = 0 \vee a_{0,2} = 0) \wedge (a_{1,0} = 0 \vee a_{1,1} = 0 \vee a_{1,2} = 0) \wedge (a_{2,0} = 0 \vee a_{2,1} = 0 \vee a_{2,2} = 0)$$

$$(a_{0,0} \neq 0 \wedge a_{0,1} \neq 0 \vee a_{0,2} \neq 0) \vee (a_{1,0} \neq 0 \wedge a_{1,1} \neq 0 \wedge a_{1,2} \neq 0) \vee (a_{2,0} \neq 0 \wedge a_{2,1} \neq 0 \wedge a_{2,2} \neq 0)$$

> The valid parenthesizations of $a_{0,0} = 0 \vee a_{0,1} = 0 \vee a_{0,2} = 0$ are $(a_{0,0} = 0 \vee a_{0,1} = 0) \vee a_{0,2} = 0$ and $a_{0,0} = 0 \vee (a_{0,1} = 0 \vee a_{0,2} = 0)$, but those always produce the same results (disjunction is **associative**) so it's unnecessary to specify which we mean. Conjunction is also associative.

Consider the following problem: **characterize** (develop an equivalent condition for) the condition that $a + b$ is **odd**, in terms of the individual values a and b being odd or **even,** using conjunction and/or disjunction. To develop an understanding of a condition people use two complementary approaches:

- gather some **necessary conditions**, i.e., restrictions that must be true for the original condition to be true; this develops a conjunction, which is complete when the restrictions narrow down the parameters to satisfy the original condition, i.e., exclude all false instances
- gather some **sufficient conditions**, i.e., cases for which the original condition is true; this develops a disjunction, which is complete when the cases are **exhaustive**, i.e., include all true instances

Determine one necessary condition, and one sufficient condition, for $a + b$ to be odd, expressed in prose (in terms of the individual values a and b being odd or even); those conditions themselves might be conjunctions or disjunctions. *Make examples as necessary for you to explore the problem.*

One sufficient condition: a is odd and b is even.
One necessary condition: a is odd or b is odd.

Determine a complete sets of necessary conditions, and a complete set of sufficient conditions, and use that to characterize when $a + b$ is odd, in two different ways.

a is odd and b is even, or a is even and b is odd
a is odd or b is odd, and a is even or b is even

Let's define $E(a)$, $E(b)$, $O(a)$, and $O(b)$ to mean a is even, b is even, a is odd, and b is odd, respectively. Use those (and $\wedge$ and $\vee$) to express the characterizations symbolically.

$(O(a) \wedge E(b)) \vee (E(a) \wedge O(b))$
$(O(a) \vee O(b)) \wedge (E(a) \vee E(b))$
Note that the first expression is a disjunction of conjunctions: the **main** / **outermost** operation is disjunction, and the entire expression is referred to as **a disjunction**. The second expression is **a conjunction** (of disjunctions).

Homework.
Use conjunction and/or disjunction to characterize each of the following (assuming $a, b, c \in \mathbb{N}$) in terms of individual conditions on their parameters:

$$\max(a, \min(b, c)) < 165$$

$$a \cdot b \geq 2$$

Include a good set of examples. Characterize each of them as conjunctions, and as disjunctions.