Problem of the Month

2022-2023

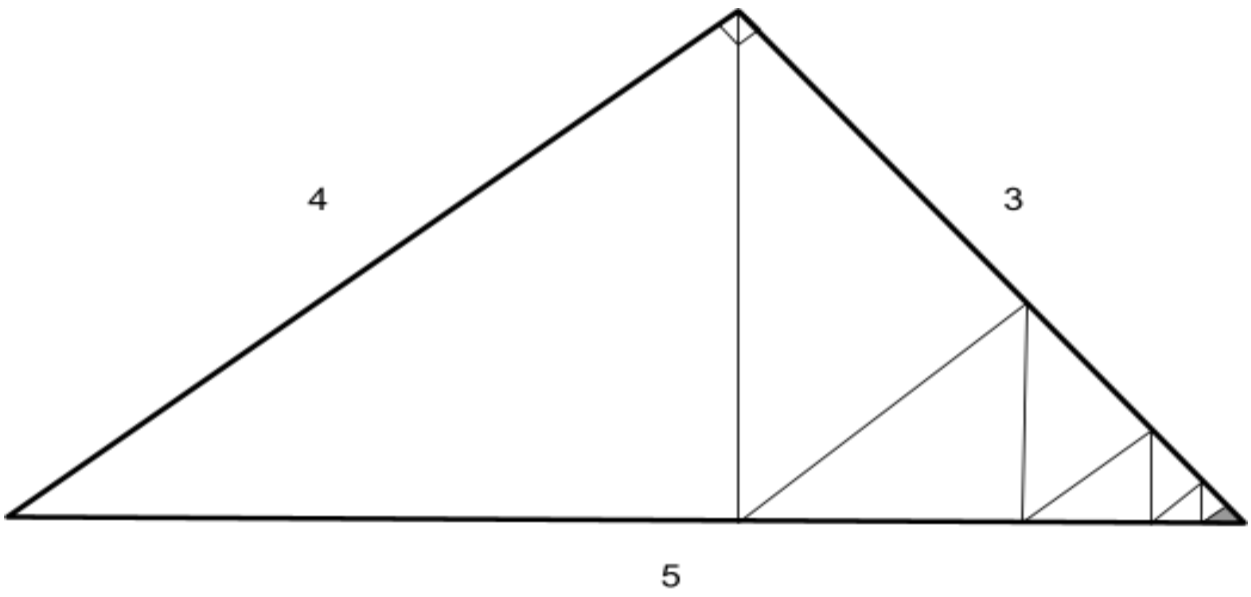
September 2022

MATHEMATICS

5 points:

A right triangle with side lengths 3, 4, 5 is drawn. Then, a line perpendicular to its hypotenuse is drawn through its right angle, resulting in two smaller triangles. This process is repeated seven additional times with one of the smaller triangles, as shown in the picture. What is the ratio of the area of the shaded triangle to the area of the original large triangle?

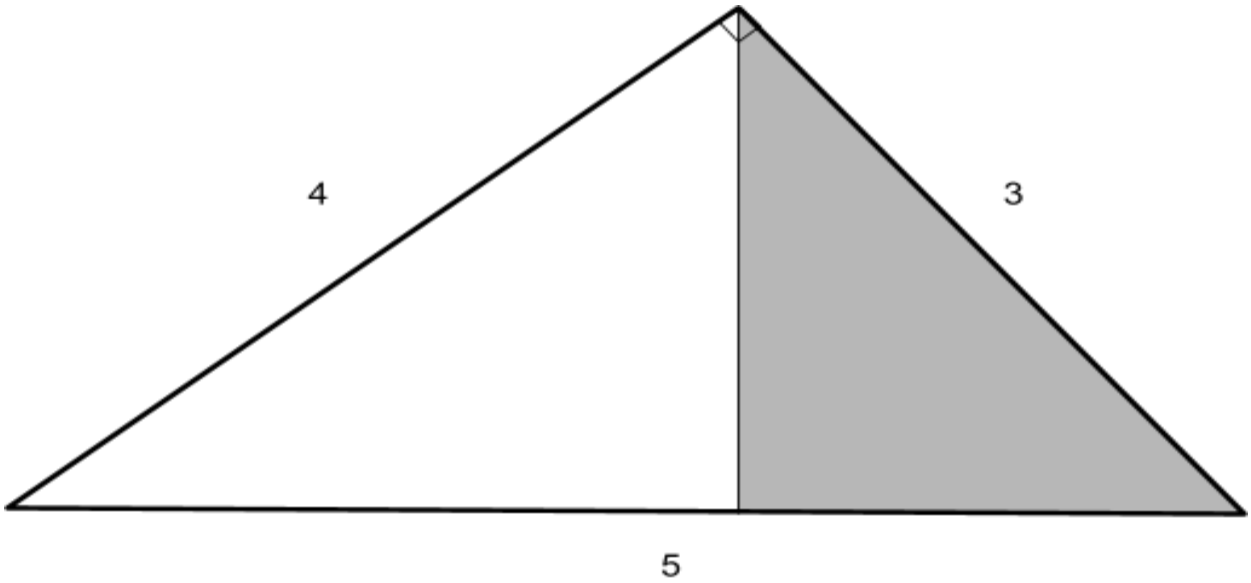
The image is not to scale.



Hint: Are some of the triangles in the image related in any special way? Could this be used to solve the problem faster?

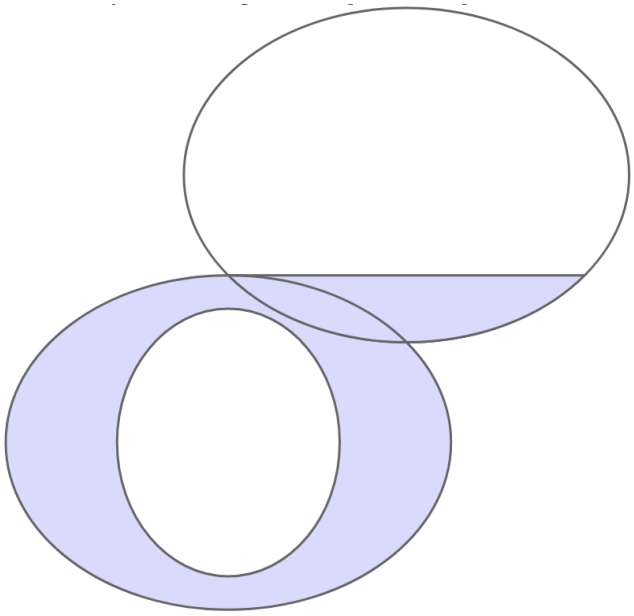
Answer: (9/25)^8

Solution:
All of the smaller triangles are similar to the largest triangle by AAA. Consider the triangle formed by the large triangle and the largest altitude drawn, illustrated below:

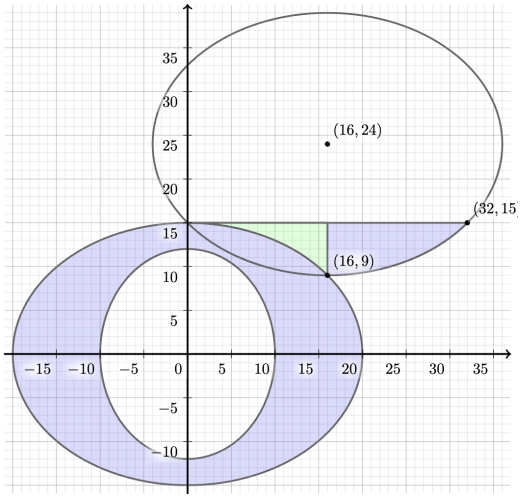


It is clear that the hypotenuse of the smaller triangle is 3, so the ratio of side lengths is 3/5. This would imply the ratio of the areas is 9/25. This “operation” is applied eight times, and the area is reduced by this factor each time, so the answer is simply (9/25)^8.

10 points:
Alice draws a lowercase sigma (σ) on graph paper using three ellipses and a line. She starts by drawing two ellipses whose semi-major axes, which run parallel to the x-axis, have length 20, and whose semi-minor axes have length 15. The first ellipse is centered at the origin, and the second is centered at (16, 24), and the ellipses have equations $(\frac{x}{20})^2 + (\frac{y}{15})^2 = 1$ and $(\frac{x-16}{20})^2 + (\frac{y-24}{15})^2 = 1$, respectively. She then draws a chord of the second ellipse which goes through (0, 15) and runs parallel to the x-axis. Finally, she draws a third ellipse centered at the origin with a semi-major axis of 12 parallel to the y-axis and a semi-minor axis of 10 parallel to the x-axis, given by the equation $(\frac{x}{10})^2 + (\frac{y}{12})^2 = 1$. Calculate the area of Alice’s sigma (the shaded area).
Don’t think you know enough about ellipses? Read up here.



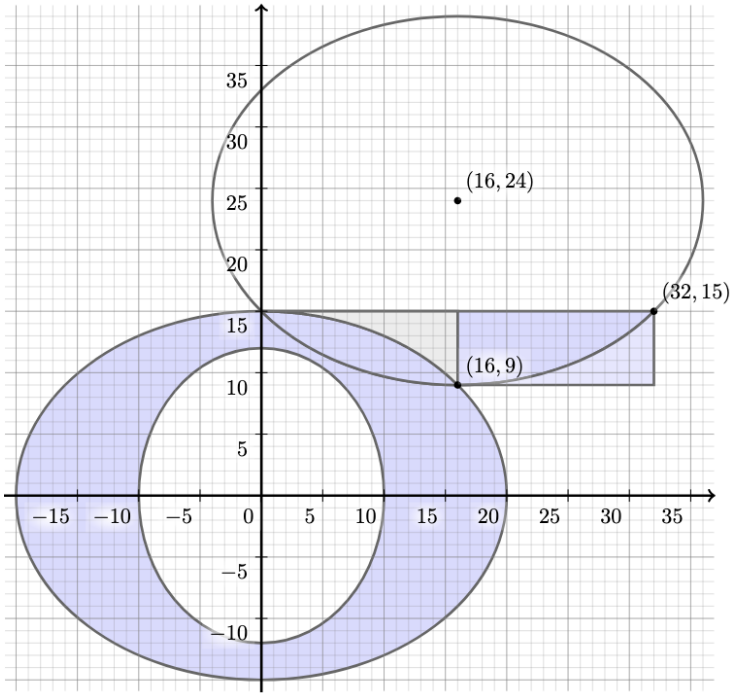
Hint:



Answer: $180\pi + 96 \approx 661.487$

Solution:

The area of Alice's sigma is the same as the area of the following figure.



The area of the outer ellipse is 20 x 15 x π = 300π and the area of the inner ellipse is 10 x 12 x π = 120π. The area of the rectangle is (32 - 16)(15 - 9) = 16 x 6 = 96. The area of Alice's sigma is the area of the outer ellipse minus the area of the inner ellipse plus the area of the rectangle, which is 300π - 120π + 96 = 180π + 96 ≈ 661.487.

PHYSICS

Preamble

Hi everyone! Welcome to Physics POM 2022-2023. There are a couple of practical points:

- **It's okay if the Problem of the Month takes a month to solve.** New concepts are hard, and one of our goals this year is to expose our contestants to much more of the vast array of cool ideas, concepts, and techniques we see across all of physics, both within and beyond what you might see at olympiads. This leads to problems that are difficult to solve if you don't know the trick, principle, or conceptual idea behind them, but fall before you quickly once you do. *So don't give up!* Each month we'll link resources and provide hints and ideas to guide you to the correct path. If you read the provided materials carefully, you'll get there!

- **We have a new side contest!** In addition to the 5pt and 10pt problems, we are introducing a third tier of problem designed to explore the material deeper and be more challenging than the main contest, for those into that sort of thing. Because we don't want to fight the other subject POM teams to the death (we *do* still need them after all) over whether Physics gets a larger share of points to hand out to contestants, we can't just create a 20pt category. **So - there will be a separate prize come Sigma 2023 for the contestant that performs best in this separate set of problems.** The prize itself is to be decided, and will be determined by the overall performance of the winner. Depending on interest, we may host an office hour after the deadline to explain the Challenge problem solution and answer any related questions. **Solutions to these problems can be intricate and require multiple lines of calculation. Please present your solution clearly and legibly, or it may be misgraded if the grader cannot follow what you are attempting to communicate.**

- **If you have any questions at all about Physics POM logistics this year, please email Alex Frenkel (this year's subject lead) at frenkelalexander1@gmail.com with the subject line `Physics POM Inquiry'.**

Introduction (Pre-ramble)

The history of the development of physics is long and full of terrors, and we don't plan to bore you by throwing a history textbook at you. That being said, while Newton formalized his laws of motion in *Principia Mathematica* in 1687 the problems presented below could very likely have been solved by ancient Greek philosophers over two thousand years ago (and here we think we've come so far since then! Don't worry too much - next month we'll graduate to at least the 18th century.)

The modern perspective, regardless, does begin with the three basic principles laid out by Newton - principles that you've seen copied and elucidated thousands of times over. Accounts of the original translation differ, and I don't know Latin, so I will trust Wikipedia's phrasing:

1. *Every body continues in its state of rest, or of uniform motion in a straight line, unless it is compelled to change that state by forces impressed upon it.*
2. *The change of motion of an object is proportional to the force impressed; and is made in the direction of the straight line in which the force is impressed.*
3. *To every action there is always opposed an equal reaction; or, the mutual actions of two bodies upon each other are always equal, and directed to contrary parts.*

There are certain folks (you know the type) that will hand these three principles to you, pat you on the back, and send you on your way - “You now have all you need to solve any mechanics problem!” While technically correct, to do this is analogous to handing someone a saw, a hammer, and a bucket of nails, asking them to turn a forest into a neighborhood, then calling them an idiot when they stand around lost and confused. The tools are there, but there is a vast wealth of tricks and techniques to using them that might appear straightforward only once they’ve been revealed to you (I’ll take any conceptual excuse to link a Penn and Teller performance). Ideas like conservation laws, normal force, kinetic and static friction, air resistance, buoyancy force, the idea that tension always points parallel to a freely hanging string, Hooke's principle, Hooke’s other principle or any of the hundred or so ideas, facts, and methods presented in this absolutely gorgeous introduction to Mechanics problem solving by Jaan Kalda (an Estonian physicist whose fantastic content we will be pulling from all year) or a similarly wonderful introduction to olympiad problem solving by Kevin Zhou (a brilliant colleague of mine at Stanford).

Even to link a barrage of (pedagogically elucidated) information, while a drastic improvement, is still quite unfair (although we highly, highly, ***highly*** recommend carefully perusing the above resources if and when you have the time). Like a chess tactic becomes infinitely easier when you know it involves a queen sacrifice, olympiad problems become much simpler once you know the idea you're meant to apply. The best olympiad solvers are those with a talent for quickly and accurately assessing which of the hundreds or thousands of tricks from their bag are the ones applicable to any particular problem. This is why for the problems presented below (and future months as well), we will not only provide resources, but *also* do our best to direct you to the needles in the conceptual haystack that are most relevant. This is our compromise for this year's spike in difficulty.

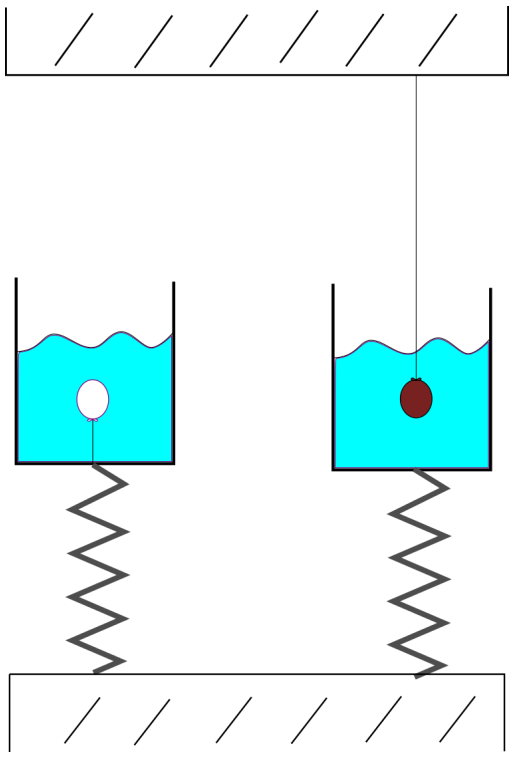
Mechanics can often feel boring due to its familiarity in everyday life. The flipside is that classical mechanics has *precisely* the tools we need to describe, explain, and predict the vast majority of everyday human experience. The curved trajectory of a spinning soccer ball, how much water you can splash at your friend by dragging your hand across a pool's surface, the structural stability of enormous arched cathedrals, or even how far you'd expect an action movie protagonist to *actually* be tossed by an explosion's shock wave - the vast majority of questions you can ask about what happens to objects in everyday life can be turned into an olympiad problem via a cleverly simplified model. Furthermore, the intuition behind energy, momentum, static and elastic collisions, and the mass flow of particles permeates all the way to every corner of cutting edge physics. More pertinently - these ideas will show up in future POMs this year, even if the monthly themes will stray to other areas of physics.

To demonstrate this last point, the mechanics problems this month are for the most part fluid dynamics themed. Fluids may seem intimidating at first, and exhibit a wealth of fascinating properties and phenomena, but one can get very far by considering a fluid as a collection of many elastically colliding particles, especially when the fluid is undergoing laminar flow (as in this month’s challenge problem!). They are a useful playground for developing your physical intuition, and practice thinking about fluids will strengthen your understanding of many phenomena you directly experience in everyday life.

So, without further ado - good luck!

5 points:

Two springs of identical spring constant support two identical tubs, filled with identical amounts of water (pictured below). The tub on the left has a balloon filled with air tied to its floor. The tub on the right holds an identical balloon, filled with an identical volume of mercury, suspended by a sturdy string from the ceiling. Provide an explanation for whether the spring on the left, the spring on the right, or neither will be compressed more. Assume the rightmost spring does not compress far enough for the mercury balloon to reach the water level in its container. **No points will be awarded for a correct answer with no solution.**



Hint: Remember Newton's third law. What counteracts buoyancy force?

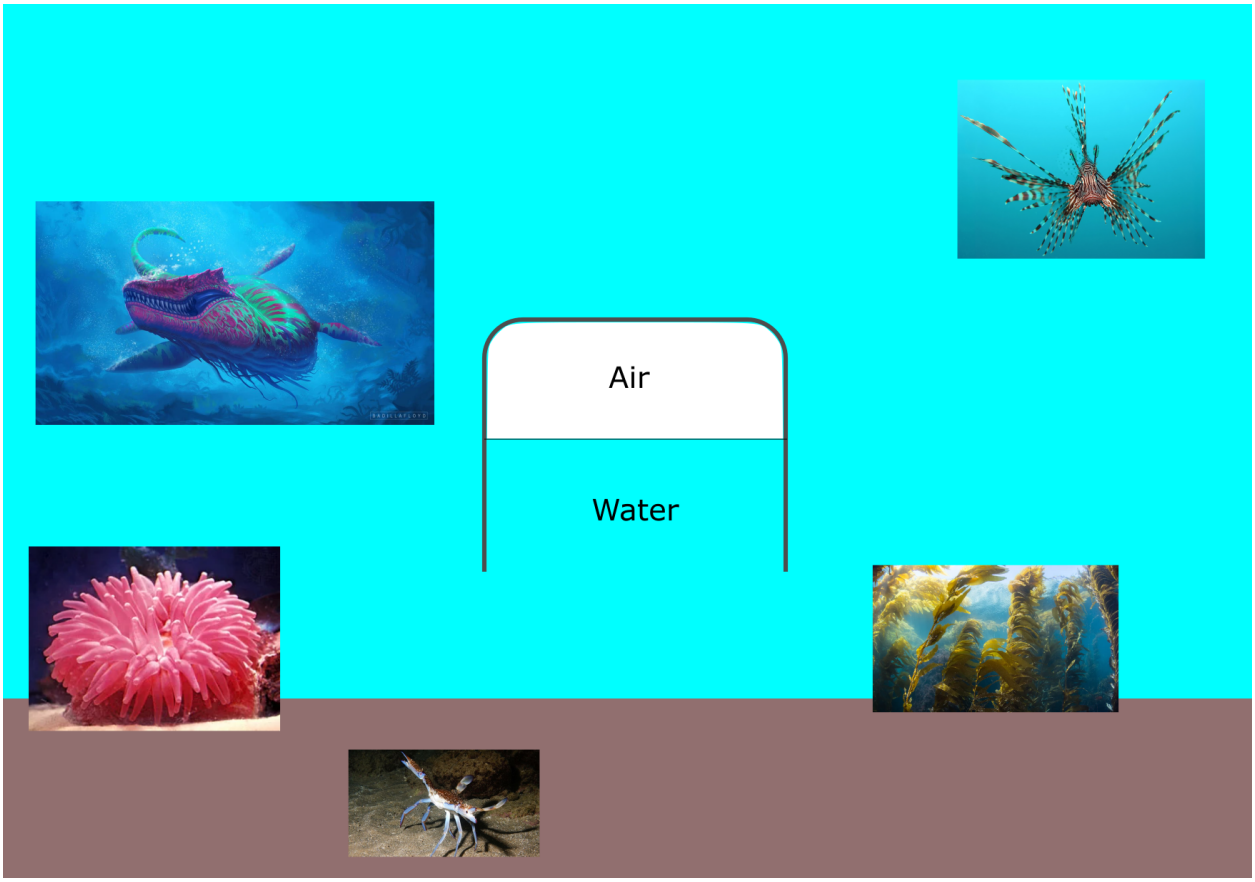
Answer: The spring beneath the lead balloon will be compressed more.

Solution: On the left, the spring is supporting the combined mass of the water and the mass of the air-filled balloon. The air-filled balloon's mass is clearly much smaller than the mass of an equal volume of water. On the right, the ceiling supports a portion of the

weight of the lead balloon. However, we know the ceiling doesn't support its *entire* weight - an object submerged in water experiences an upward buoyancy force, proportional to its volume times the density of water. By Newton's third law, we know this upward force must generate an additional downward force on the spring, equal to the density of water multiplied by the mass of the balloon

10 points:

For our second problem, we place a bucket of mass M resides in placid, idyllic water. It floats at depth D, supported by an air pocket trapped inside. At this depth, all forces acting on the bucket cancel. A passing megalodon disturbs the bucket, causing it to begin floating upward. At what depth will its acceleration reach $1g$? Assume the initial volume of this air pocket is V_0, and the bucket is large enough that at no point does air spill out of the bottom.



Hint: As the bucket rises, the pocket of air expands due to the lowering pressure, and the effective amount of displaced water due to the air changes.

Answer: The depth at which the bucket reaches acceleration 1g will be $D' = DV_0(V_0 + M/\rho)^{-1}$.

Solution: At depth D below sea level, the ambient pressure is $P = \rho g D$, where ρ is the density of water. The ideal gas law tells us that the volume $V \propto P^{-1}$. As the bucket rises, the air contained inside expands due to the decreasing pressure, and the total volume of the bucket and the air *increases.* From the pressure-depth relationship, we can read off that $V \propto D^{-1}$.

The bucket's acceleration upward will equal g when the buoyancy force acting on it increases from its initial configuration by Mg. We therefore need the volume of the air to decrease by an amount ΔV such that $\Delta V \rho g = Mg \rightarrow \Delta V \rho = M \rightarrow \Delta V = M/\rho$. The *ratio* of the new volume to the initial volume will be $(V_0 + \Delta V)/V_0 = (V_0 + M/\rho) / V_0$. Due to the inverse relationship between depth and volume, this ratio will also be the ratio of the *initial* depth to the *new* depth. As a sanity check, the new depth is indeed smaller than the initial depth.

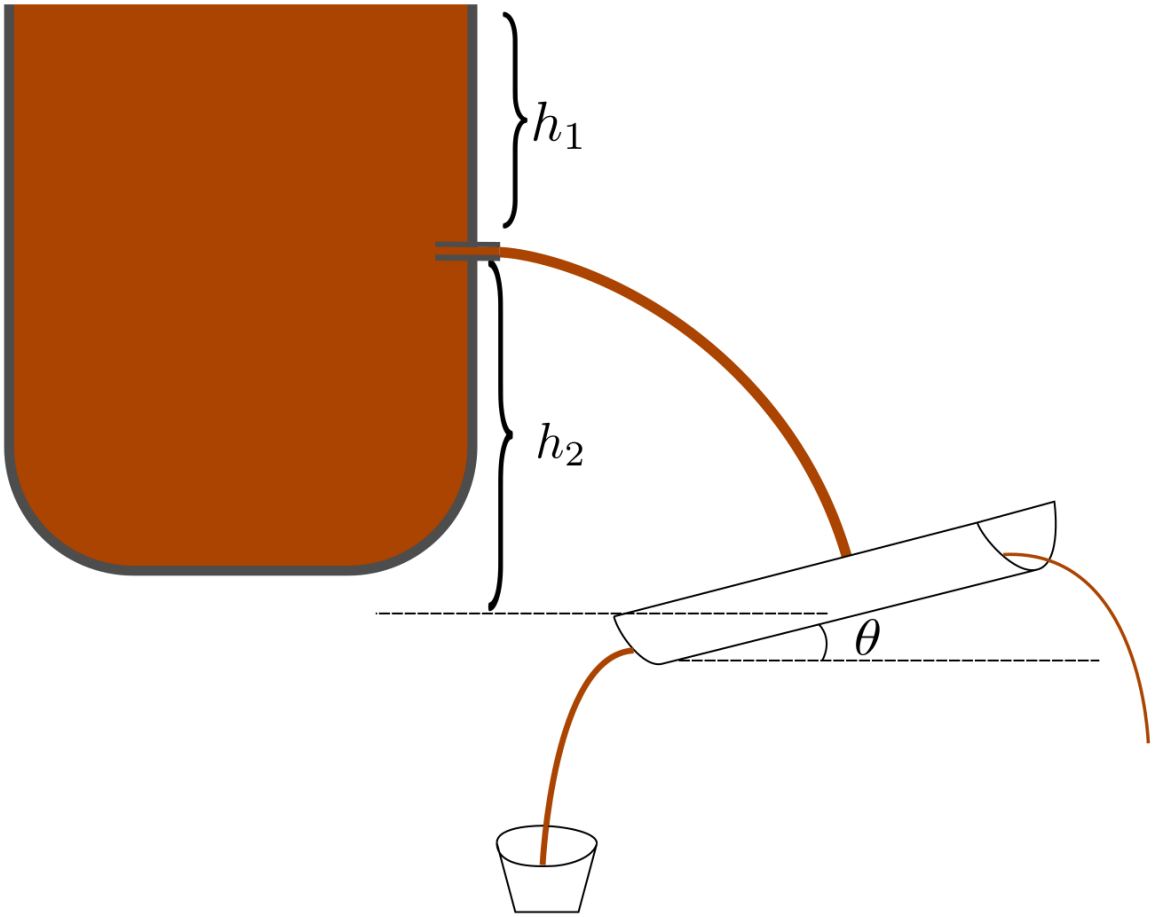
September Physics Challenge Problem

Warmup: Problem 52 of Kalda. Understanding and submitting the solution to the warmup is worth 10% of the total score this month.

The main event: A bored Sigma camper has drilled a hole in their grandmother's expensive samovar! The camper proceeds to fill their cup by letting the tea pour into a gutter, which then redirects it into their cup. However - the camper forgot to make the gutter long enough, and some of the tea is spilling over the top. How much tea is entering the camper's cup per second?

Assume the hole is drilled at height h_1 below the tea level in the samovar, the tea hits the pipe at height h_2 below the hole, the pipe is inclined at angle θ, the tea undergoes laminar

flow, you can neglect air resistance, and the pipe is short enough that all the tea that is directed up the gutter after collision spills out the top. Use ρ for the density of the tea.



CHEMISTRY

5 points:

Alice built a mini hydroponic farm in her house, and decided to grow tomatoes there. She prepared a solution containing 10 mM ammonium chloride and 10 mM potassium dihydrophosphate, which was supposed to provide tomatoes with the three most essential elements: potassium, nitrogen, and phosphorus. However, she quickly found that tomatoes were growing very slowly, and she asked Bob for advice. "Check the pH", he said, "It seems your solution is too acidic." Alice quickly realized that he was right, and she decided to fix that problem by adding lime water (a solution of calcium hydroxide) until pH reached the value of 7.4. After that, Alice let the solution obtained sit for one day to let all small particles precipitate, and then she used the clear liquid for growing tomatoes. However, although tomatoes did grow, their leaves were dark, dull, blue-green, and may become pale in severe deficiency. Alice again asked Bob's help, and he immediately found the reason. "Alice, you forgot that ....." What did Bob say?

Hint:

Solution:
Potassium dihydrodenphosphate is an acid salt, so it is moderately acid: Its formula is KH_2PO_4, which means it has one acidic hydroxide that needs to be neutralized to get K_2HPO_4 (the salt that is only slightly basic). That can be achieved by adding some alkali, and calcium hydroxide is sufficiently basic for that purpose. However, one product of that reaction is virtually insoluble. This product is calcium monohydrogen phosphate ($CaHPO_4$):

$$NH_4Cl + KH_2PO_4 + Ca(OH)_2 \rightarrow CaHPO_4 + 2H_2O + KCl + NH_3$$

Since calcium hydorgenphosphate precipitates, the amount of soluble phosphorus decreases, which causes phosphate starvation. Therefore, most likely, Bod reminded Alice about poor solubility of calcium phosphates.

10 points:

Consider this statement: "In some geographical location, the following phenomenae are observed: (i) The water from natural wells has a strong H_2S smell; (ii) when you bite a local apple and leave it for several minutes, the surface becomes brown; (iii) teapots become covered with a thick scale when they have been extensively used during several days; (iv) when you are pumping fresh water from local wells, it is initially clear and colourless, but if you leave it in a bucket, it quickly becomes turbid; (v) the surrounding area swampy and wooded; (vi) the location is surrounded with red sand dunes." Identify which items in the list correlate with each other and which are mutually exclusive. Explain your answer from a chemist's point of view.

Hint:

Solution:

The chemistry behind these phenomenae is as follows:
(i) Hydrogen sulfide forms as a result of activity of sulfur reducing bacteria, and that happens in anaerobic conditions. Normally, when oxygen is present, there is no need to use other oxidizers, because reduction of oxygen is a much more efficient process.
(ii) One popular misconception is that apples become brown after you bite or cut them when the apples are rich in iron. Indeed, most common oxidation states of iron are II and III, and ferrous salts (for example, ferrous chloride, $FeCl_2$) are pale green, and they are almost colourless at low concentrations. In contrast, ferric salts (for example, ferric chloride, $FeCl_3$) are dark brown. Importantly, oxidation of ferrous salts to ferric salts in the presence of atmospheric oxygen occurs very easily, and if, for example, you leave a dilute solution of $FeCl_2$ in an open beaker on a table, it quickly becomes brownish and turbid because of oxidation of $FeCl_2$:

$$4\ FeCl_2 + O_2 + 3\ H_2O \rightarrow 4\ FeCl_2OH$$

The basic ferric chloride is brown, and it further hydrolyzed to brown ferric hydroxide (which is insoluble) and dilute HCl.

However, that does NOT happen in apples (contrary to what some old books say) just because there is not enough iron in apples. In contrast, that is exactly what happens in “(iv)”, see below.

Actually, cut apples become brown for a totally different reason, and the chemistry of that process is close to the transformations that occur when green teal leaves become processed to bake black tea.
(iii) The scale that forms in teapots is calcium and magnesium carbonate. How do they form? They form when tap water is hard. “Hardness of water” means high concentration of calcium and magnesium salts, usually hydrocarbonates and chlorides. When hard water is boiled, nothing happens to calcium and magnesium chlorides, but their hydrocarbonates decompose. For example, calcium hydrocarbonate ($Ca(HCO_3)_2$) decomposes according to the equation:

$Ca(HCO_3)_2 \rightarrow CaCO_3 + CO_2 + H_2O$

In contrast to hydrocarbonates, carbonates of calcium and magnesium are insoluble, so they precipitate out, and thick scale quickly forms in a teapot.
(iv) As we already explained, if a water pumped from a well is initially clear and colourless, but it becomes brownish upon standing, that happens because of the presence of ferric salts there. Where ferric salts (a.k.a divalent iron salts) come from? Obviously, that happens when ground water passes through some iron rich sediments. The problem is, however, that most iron in the Earth crust is in trivalent form, and trivalent iron is usually poorly soluble in neutral media. Example is red sand, which is red because of the presence of iron (see below). How can trivalent iron be converted into divalent one? Obviously, that can be done via the process called reduction (reduction is the process opposite to oxidation). That may happen only in anoxic conditions (no oxygen), and that is usually a result of activity of some anaerobic bacteria. In addition, to solubilise iron, it is desirable to have slightly acidic media. These conditions are met in swampy and wooded lands.
(v) In swamps, dead organic matter sinks, and its decay occurs in the absence of oxygen, so the products are turf, hydrogen sulfide, humic acids, etc., so the media is highly reducing. If a soil is rich in trivalent iron (for example, it contains red sand) the iron becomes reduced. By the way, if ground water from swamps come to a contact with air, the dissolved iron is reoxidized, and precipitates on the bottom in a form of so called “bog iron ore”, which was the main source of iron in Medieval Eastern Europe and New England at the beginning of its colonization.
(vi) Red colour of sand dunes is usually due to the presence of ferric oxide (Fe_2O_3).

From that, we can easily see that phenomenae (i), (iv), and (v) correlate with each other, (vi) also may have some relationship to them, although it is hard to imagine coexistence of sand dunes and swamps, whereas (ii) has no relation to them. In addition, usually,

ground water in swampy lands is soft, so correlation of (iii) with (i), (iv), and (v) is unlikely.

BIOLOGY

Introduction

Welcome to the Biology POM 2022-2023! We're very excited to introduce you to the world of scientific literature through the POMs this year, and in that vein, here are some tips to approach tackling a scientific article. When answering questions, we expect you to do research and consult outside sources, but you do need to cite every source you use in a consistent format (MLA 9th Edition). Here are some resources to help you with that! As always, don't hesitate to contact Elena Yakubovskaya (elena.yakub@gmail.com) or Sanjana Rao (sanjanarao@uchicago.edu), this year's Biology subject leads, if you have any questions.

5 points: *Humans have subjected many animals to artificial selection, creating breeds to better fit humanity needs. Provide several examples of such animals and the purpose for which they are bred. Why do you think there are many more examples of vertebrate animals domesticated in this way, but only few among the invertebrates? What difficulties may humans encounter when trying to domesticate invertebrates? Suggest one invertebrate species that would be useful for humans to create an enhanced breed for and describe your artificial selection strategy.*

Hint:

Answer: Dogs for protection and hunting, horses for transportation, cows for milk, sheep for wool, silkworm for silk, chicken for eggs, cats to keep away the mice etc. Examples like oysters/shrimps cultivated for food do not fit because they were not bred to enhance a specific trait, they were just captured and cultivated for mass production. Bees are still controversial - some scientists say they are not "domesticated" but "managed" by humans, because there are not many genetic differences with their wild counterparts (https://www.intechopen.com/chapters/64219).
Invertebrates are small, hard to manipulate and contain. They are not too useful: no wool, not much meat, not fit for transportation/hauling things. The most common vector for "domestication" was a selection of people-friendly animals and the formation of whatever creature that is not afraid of humans. This is possible to do in vertebrates since they have a more sophisticated nervous system, but not possible with most of the invertebrates.

10 points: *Poaching is a serious threat to the wild black rhinoceros in southern Africa. Governments are actively managing the black rhinoceros population across southern Africa, mostly by capturing and subsequent translocation by a helicopter. Rhinoceroses are transported in suspension in two common positions: lying on the side and upside down, hanging by their feet [1]. Which way is safer for a rhinoceros's health and why? How is the safest transportation orientation different for other big mammals? Provide examples. How would you set up an experiment to test the best orientation for transportation?*

(The Pulmonary and Metabolic Effects of Suspension by the Feet Compared with Lateral Recumbency in Immobilized Black Rhinoceroses (Diceros bicornis) Captured by Aerial Darting)[1]

Hint: no hints

Solution: According to the paper "Established protocols for capture of wild black rhinoceroses use potent opioids", and one of the well-known side-effects of opioid overdose is respiratory depression. Thus, rhinos may have problems breathing during the transportation, which may lead to hypoxemia (low blood-levels of oxygen), which can have serious side effects such as brain damage and death. The paper mentions that black
rhinoceros in sternal recumbency (normal orientation) have significantly greater median arterial oxygen pressure - meaning less chance of hypoxemia. However, that method of transportation is not widespread, because it is harder to reliably suspend a rhino in that position. Out of the two remaining orientations, according to the paper, **suspension by**

[1] *Radcliffe et al. Journal of Wildlife Diseases, 2021.*

feet had numerically small, but statistically significant greater mean arterial oxygen pressure and lower mean arterial carbon dioxide pressure, indicating better respiratory gas exchange. The gas pressure differences, albeit small, are clinically significant in an animal that is already hypoxic from the opioids. The hypothesis explaining this difference according to the paper is as follows: in rhinos laying on the side the blood from the top lung accumulates in the bottom lung, resulting in low blood perfusion in the top lung, leading to increased “alveolar dead space” - portion of the air that was exhaled before gas exchange with the blood.

According to the paper, both white rhinoceros and horse exhibited better gas exchange in **lateral position** in contrast with black rhinoceros.

Suggested experiment set up: use darts with potent opioids to sedate 20 species of interest of various ages and sexes, of **similar weight** (to have the same opioid dose per kg). Split the group of test subjects in 2 groups of 10 with equal age and sex compositions. Put the first group in lateral recumbency and suspend the second group by the feet for 30 minutes (use longer time to account for actual transportation times). After 30 minutes, collect expired air from the nostrils using cuffed tubes and a plastic bag for 2 minutes counting the breathing rate, then collect arterial blood right after. Next, switch the position of each animal - lateral to dorsal and vice versa - wait for 30 minutes and collect the same measurements. This is the minimal amount of measurements needed - now you can compare arterial carbon dioxide and oxygen pressures for each animal in 2 different positions. To draw conclusions about best transportation methods you need to see consistent results: arterial oxygen pressure is higher and arterial CO2 pressure is lower in the same posture. Use a one-tailed t-test to check your hypothesis. The resulting p-value of less than 0.05 indicates significant differences in gas exchange in the two different postures.

To expand the dataset and account for time since the opioid injection you may keep changing the animals positions and record the data until the animal starts to wake. That will allow to track changes in the arterial gas pressure differences with time.

LINGUISTICS

5 points:

An anagram is a word or phrase that can be rearranged to create another word or phrase. An aptigram is an anagram that creates a synonym when rearranged, and an antigram creates an antonym. Given these three definitions, find an existing prefix used in English (like apti- or anti-) and create your own type of anagram. Provide a definition and three examples of your new type of anagram. The examples, both before and after rearranging, must use distinct valid English words, and proper nouns are not permitted.

Example: aptigram (apti- from Latin, meaning fitness)
past due = date's up
dormitory = dirty room
a gentleman = elegant man

Hint:

No hint this month!

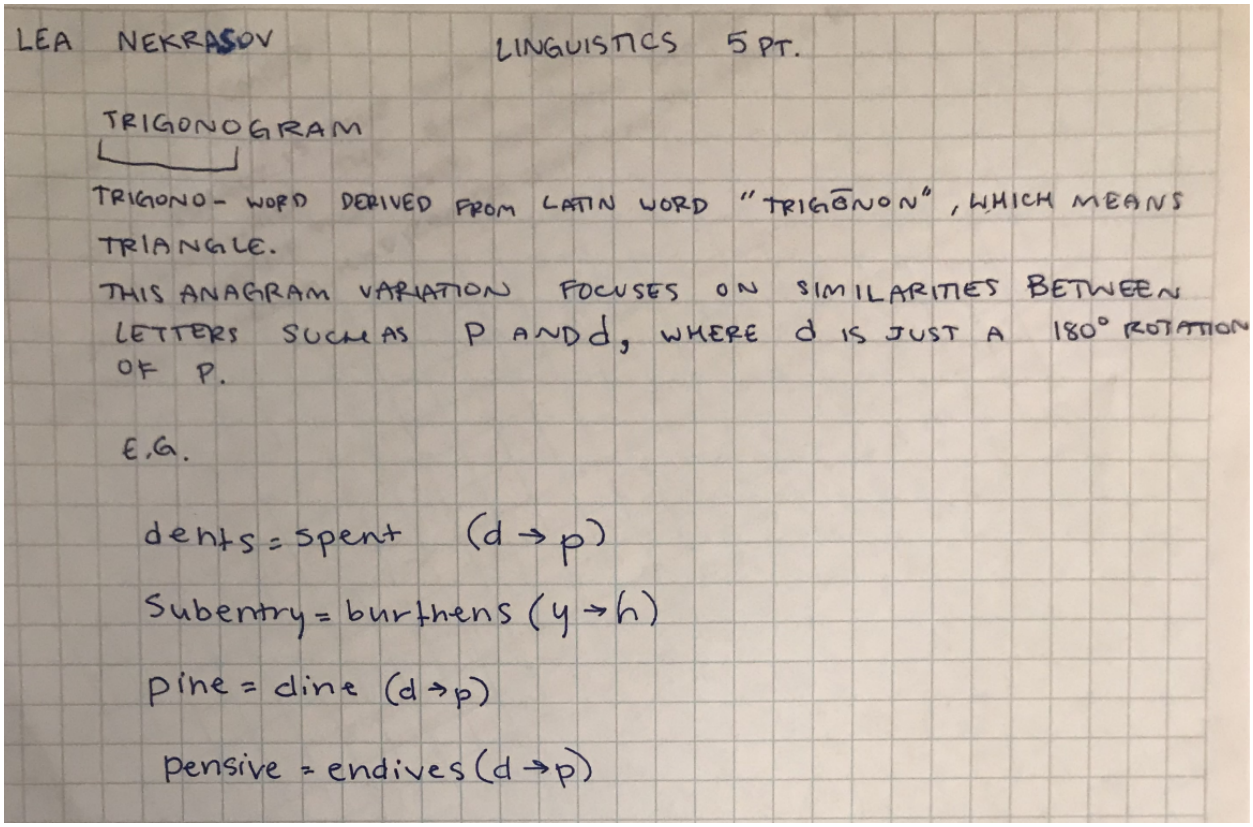
Answer:

There is no one right answer, but as described in the problem for full points the solution must a) use an existing prefix used in English b) use only valid English words c) the examples must be valid anagrams d) the examples must match the definition of the new anagram type.

Solution:

Here are two submitted solutions that we chose to be featured this month:

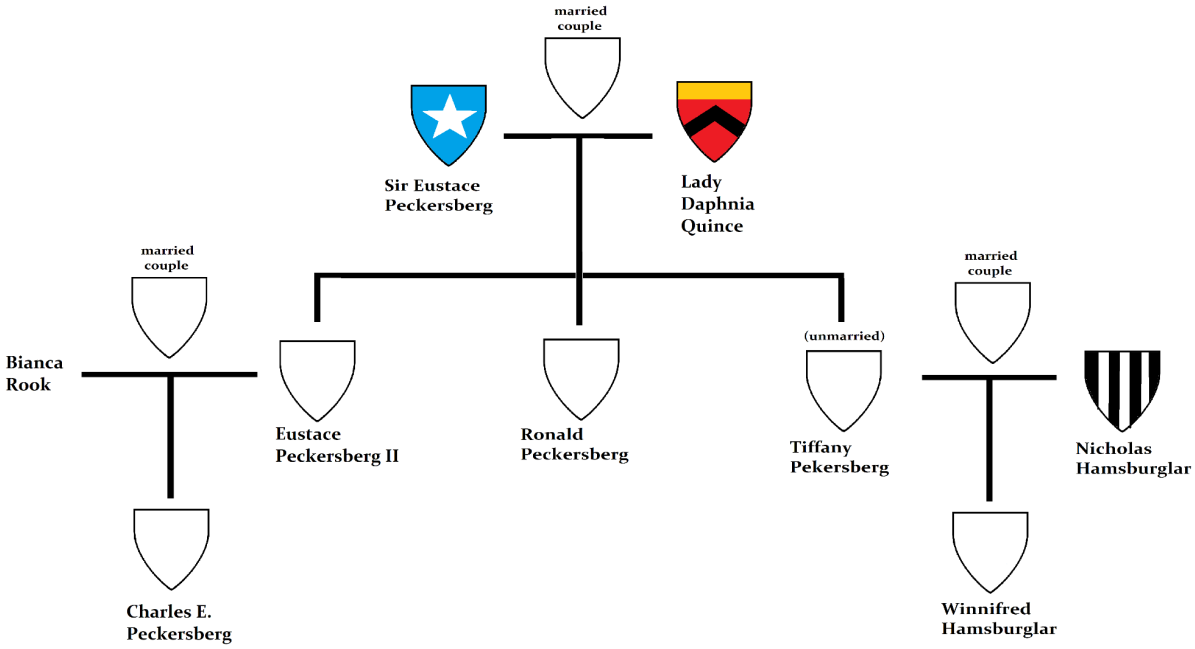
By Lea Nekrasov:



10 points:

Heraldry is the broad term for things related to the study and design of heraldic achievements, most commonly known as coats of arms. Coats of arms are passed down through families, but change as they are passed down to different family members.
Below is a family tree with only three of the coats of arms provided. Please research the rules for passing down coats of arms and add your own illustrations to the blank spaces in the tree.

Note: We understand you might find conflicting sources, so please explain why you choose to draw each coat of arms that way and cite your sources. Please do your best to be clear with your drawings, but if you're worried that they aren't clear you can provide a description next to them. The diagram below can be downloaded here: https://drive.google.com/file/d/1szQOeKiHpjcvRXiDoGJ-N6vOsv71_zhC/view?usp=sharing



Hint:

You're looking for rules about:
- combining coats of arms after a marriage
- symbols added to coats of arms based on children's birth order
- women inheriting coats of arms

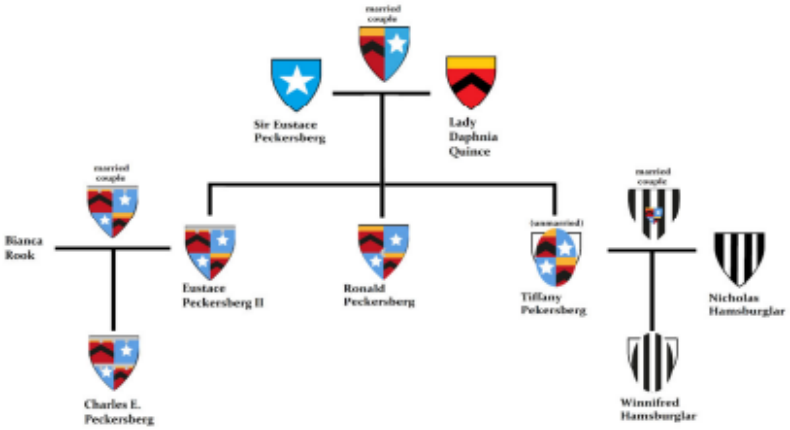
Answer:

Due to the differences in heraldric tradition between countries and

Solution:

Here are two submitted solutions we chose to be featured this month:

By Alice Goward:



These all follow the UK college of arms government site laws, which are very similar but possibly not the same within all details as the generally accepted US ones. By these laws, Lady Daphnia Quince should have a oval or lozenge(diamond) coat of arms, as women have these shapes rather than a shield, which is reserved for married couples or men.

Given that Lady Daphnia Quince has a title, it is presumed she is a heraldic heiress (she is the oldest sister and has no brothers). This means that the crest of the married couple has them both equally, side by side. The man's coat of arms is usually on the right.

Eustace Peckersberg II has a quartered shield, as both his parents pass down their coats of arms. The coat of arms passed from the father usually goes in the top right and bottom left, while the mother's in the top left and bottom right. Due to being the oldest son, he also has a horizontal strip with three pendant drops (shown in gray)

The married couple coat of arms between Eustace Peckersberg II and Bianca Rook is the same as Eustace Peckersberg II's, as Bianca Rook does not have a coat of arms of her own.

Charles E Peckersberg is the same as his father's, although due to being the first son another horizontal strip with three pendant drops is added

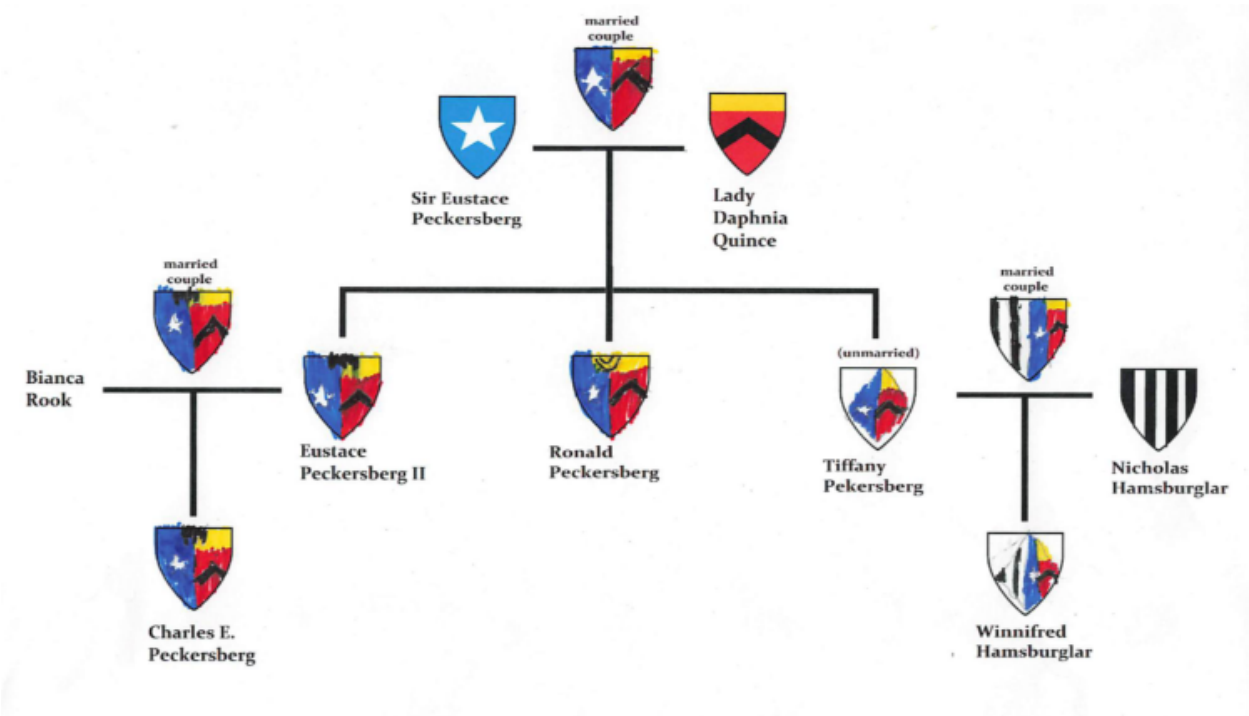
Ronald Peckersberg has a very similar coat of arms to his brother, but due to being the second son he has a crescent instead of the horizontal strip with three pendant drops.

Tiffany Peckersberg is not a heraldic heiress due to having brothers but is still able to use the coat of arms, just not pass it on. However, due to her being female it must be in either a lozenge (diamond) or oval shape.

The married couple coat of arms between Nicholas Hamsburglar and Tiffany Peckersberg is the same as Nicholas Hamsburglar's, but with Tiffany's coat of arms as an inescutcheon of pretence on a shield (very small, in the center of the coat of arms), as she has a family coat of arms but is not a heraldic heiress.

As Tiffany Peckersberg's coat of arms does not pass on to her daughter, Winnifred Hamsburglar only gets Nicholas Hamsburglar's coat of arms. As she is female, it is on an oval, but as she does not have any brothers she is a heraldic heiress.

By Tyler Malkin:



This is my illustration of the family tree. I impaled the shields of Sir Eustace and Lade Daphina Quince (with Sir Eustace on the dexter and Lady Daphina on the sinister). Their children each inherited the married coat of arms with certain distinctions. The first-born and second-born son each carry an extra mark of cadency. For the first-born son, it is a label of 3 points. For the second-born son, it is a crescent. Their daughter Tiffany carries the married coat of arms on a lozenge (diamond shape), rather than filling a crest. I impaled Tiffany and Nicolas coat of arms to create their married couple shield (with Nicolas on the dexter and Tiffany on the sinister). Then, this shield was passed onto their daughter Winnifred, but she carries it on a lozenge. As for Eustace II (the first-born son), his shield is also used as the married couple shield (as his wife Bianca did not have a coat of arms of her own) and was inherited by their first-born son Charles.

I used several sources to create my illustration including:
https://www.fleurdelis.com/marshalling.htm

https://www.theheraldrysociety.com/articles/the-arms-of-women-a-decree/#:~:text=2.,husband%20in%20the%20normal%20way

https://www.britannica.com/topic/heraldry/Manipulation-of-heraldic-design

COMPUTER SCIENCE

While computer science is a lot of fun (we hope!), it can also be complicated and difficult to learn. We want these problems to be challenging but approachable, so this year's CS POM is introducing some new resources to make learning the skills to succeed more accessible:

- This year, the CS POM team has decided to introduce a lecture for September, which will consist of three 20-minute time blocks, during which three sessions will be offered concurrently. These beginner-friendly sessions will be taught by members of the CS POM team, and will cover a wide range of CS topics, mostly centering around Python. If you haven't coded before, we suggest making an account with Replit, a website that can run your code, which we'll use during the sessions. The lecture will be offered on **Sunday, October 2nd at 3PM EST** at https://sigmacamp-org.zoom.us/j/88483055125. Can't make it? We will record all the sessions and post links to the recordings as well as notes from some sessions on the POM website. Participants can attend as many or as few of the time blocks as they want - it's up to you to decide which sessions you'd like to attend! Depending on the lecture's success, it may be offered again in future months with some new topics. The schedule of sessions is below:

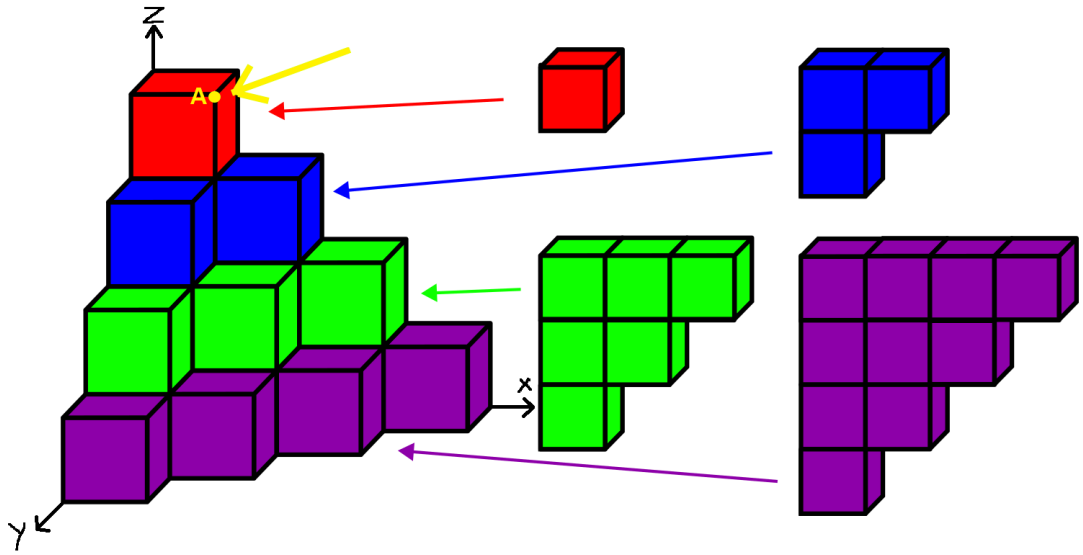
Introduction	
3:05 - 3:25 PM EST	Get Started with Python with Anna Rosner
	Read and Write a File in Python with Alexey Tatarinov
	Python Logic Basics with Anatoly Zavyalov
5 minute break	
3:30 - 3:50 PM EST	Data Processing and Validation with Anna Rosner
	Read and Write a File in Python with Alexey Tatarinov
	Python Logic Basics with Anatoly Zavyalov
5 minute break	
3:55 - 4:15 PM EST	Data Processing and Validation with Anna Rosner
	Python Data Structures with Anar Amgalan
	Depth-First Search and Breadth-First Search with Anatoly Zavyalov

- Additionally, the CS POM team has put together a list of some of their favorite CS resources. Any that we feel are relevant to a problem will be linked within it, but otherwise you can find the whole list here. Without further ado, here are the Computer Science rules and problems:

- Your program should be written either in Java or Python 3
- No GUI should be used in your program
- All the input and output should be done through files named as specified in the problem statement
- Java programs should be submitted in a file with extension .java; Python 3 programs should be submitted in a file with extension .py.
  No .txt, .dat, .pdf, .doc, .docx, etc. Programs submitted in incorrect format will not receive any points!

5 points:

The picture below shows a pyramid built out of identical cubes. The base of the pyramid is a right triangle, and the pyramid is built up along the z-axis. The top of the pyramid has 1 cube, the second layer from the top has 3 cubes, the next layer has 6 cubes, etc. For example, the picture below shows a 4-layer pyramid:



Write a program that receives the coordinates of the outermost point of the top cube (A, pointed to by the yellow arrow in the diagram above) and calculates the total number of cubes in the pyramid. Your program should read the input file **input.txt**, which will consist of one line containing the x, y, and z coordinates separated by spaces.

Example input file:

```
1 1 4
```

Your program should produce the output file **output.txt**, containing a single integer number representing the volume of the pyramid (i. e., total volume of all the cubes comprising the pyramid). If the given input does not correspond to a possible pyramid, write IMPOSSIBLE to the output file.

Note that the provided coordinates are integer numbers, and that **the size of each cube is not necessarily 1x1x1**.

Hint:

No hint this month!

Solution:

We realized there was some ambiguity over whether programs should output the volume of the pyramid or the number of cubes it contains, but programs should have outputted the total volume, as was specified above. Please do not hesitate to ask any questions if there is any uncertainty about a question, whether that is in the Discord or via emailing pom@sigmacamp.org.

Python:

```
with open('input.txt') as f:
	lines = f.read()
	# uses a list comprehension to assign dimension variables
	x, y, z = [int(n) for n in lines.split()]

	# check that they're actually cubes and fit within the height
	if (x == y and z % x == 0):
	# how big each cube is
	size = x
	volume = x**3
	# update x, y, z to make the size of each cube 1
	x, y, z = x // size, y // size, z // size
	# total number of cubes so far
	sum = 0
	# the number of cubes in the previous layer
	prev = 0

	# this essentially calculates triangle numbers
	# the number of cubes in the nth layer is equal to the nth triangle number
	# you can also use a formula to calculate this sum, found here:
```

```
	#https://math.stackexchange.com/questions/2435816/a-formula-for-the-sum-of-the-t
	riangular-numbers

	for i in range(z):
		prev += i + 1
		sum += prev
	string_to_write = str(sum*volume)
	else:
	string_to_write = "IMPOSSIBLE"

	out = open("output.txt", "w")
	out.write(string_to_write)
	out.close()
```

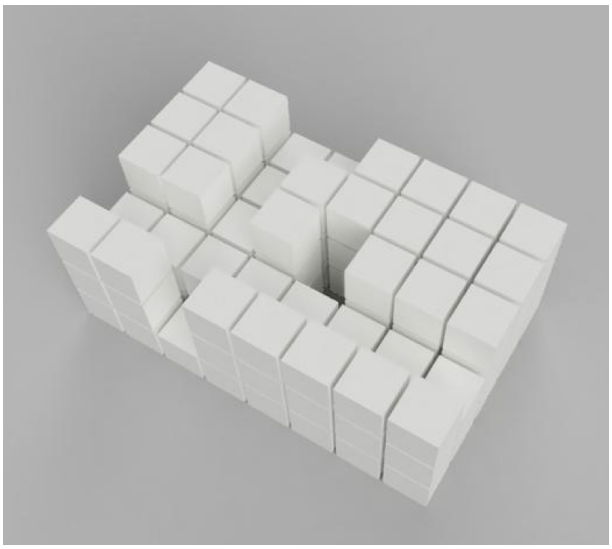
Java:

```
import java.io.File;
import java.util.Scanner;
import java.io.FileWriter;
import java.io.IOException;

class Main {
	public static void main(String[] args) {
	try {
		// make a file and a scanner to read the file
		File file = new File("input.txt");
		Scanner sc = new Scanner(file);

		// empty coords array
		String[] coords;
		while (sc.hasNextLine()) {
		// read the line, then split it by spaces ("\\s+" means one or more spaces
in regex)
		coords = sc.nextLine().split("\\s+");
		int x = Integer.parseInt(coords[0]),
			y = Integer.parseInt(coords[1]),
			z = Integer.parseInt(coords[2]);

		String stringToWrite = "";
		if (x == y && z % x == 0) {

			// how big the size of each cube is
			int size = x;
			int volume = Math.pow(x, 3);

			// update x, y, z to make the size of each cube 1
			x = x / size;
			y = y / size;
```

```
                        z = z / size;

                        // the total number of cubes so far
                        int sum = 0;

                        // prev is the number of cubes in the previous layer
                        int prev = 0;

                        for (int i = 0; i < z; i++) {
                               prev += i + 1;
                               sum += prev;
                        }
                        stringToWrite = Integer.toString(sum * volume);
                  }
                  else {
                        stringToWrite = "IMPOSSIBLE";
                  }

                  // write to the output file
                  FileWriter out = new FileWriter("output.txt");
                  out.write(stringToWrite);
                  out.close();

                  }
                  sc.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
            }
}
```

10 points:

A three-dimensional figure is constructed from identical cubes with overall dimensions l x w x h cubes. All the cubes are aligned along the grid lines with no displacement. From this figure, projections are made from three sides: front, right, and top. To make a projection of a figure from a view, look at it straight on. For each cell where any cube is visible, that cell is shaded. If no cube is visible, that cell is empty.
For example, the projections of the below figure would be the following:



```
## #####
######## (from the front)
########

# ###
##### (from the right)
#####

########
########
#### ### (from the top)
########
########
```

Using these projections and given values for l, w, and h, your program will calculate the maximum number of cubes that could be in the figure or identify if such a figure is impossible (eg, projections are inconsistent). Your program should read the input file **input.txt**, of which the first line will include l, w, and h, each separated by spaces, and the following lines will include the projections, each separated by an empty line. In the projections shaded cells are represented by #.

Example input file:

```
3 4 3
####
####
# ##

###
###
###
```

```
####
 ###
####
```

Your program should produce the output file **output.txt**, containing a single integer number representing the maximum number of the cubes in the figure to satisfy given projections. If the projections given do not correspond to any possible figure, write IMPOSSIBLE to the output file.

Hint:

No hint this month!

Solution:

We realized that we were not very clear on the order of the projections in the input. We intended for the projections to be in any order, but in the problem statement implied an order of front, right, top. We will present a solution that can handle any order, but we will not penalize for assuming the order of the projections.

Python:

```
# There is no one way to solve this problem, that's what makes it interesting! This is
# just the way we did it.
# We use list comprehensions a bit in this solution, and they're really useful! You
# can learn more about them here:
# https://www.w3schools.com/python/python_lists_comprehension.asp
# This solution can definitely be optimized, we chose to prioritize making it as
# understandable as possible.

possible = True #by default, the projection is possible
with open("input.txt","r") as input:
   #read the first line and split into a list, assigning dimension variables
   dimensions = input.readline().split()
   projs = input.read()

if len(dimensions) != 3:
   possible = False

if possible:
   l, w, h = [int(dim) for dim in dimensions]
   #these are the valid dimensions a projection can have
   validprojs = [[l, h],[w, h],[l, w]]
    #it's important to split by only a blank line, or you risk eliminating intentional
    # whitespace.
   projs = projs.rstrip("\n").split("\n\n")
   #split each projection into a list of lines
```

```
   projlist = [proj.split("\n") for proj in projs]

# Check to make sure one projection is l x w, one is w x h, and one is l x h.
# Otherwise, it's impossible.
# also orders the projections in front, side, top order, after which the projections
# can be solved as if they were given in that order to begin with
if possible:
   projSort = []
   for dims in validprojs:
       # this is a great way to index through a list while also keeping track of where
       # you are!
       for index, proj in enumerate(projlist):
           #checks that height and length of projection are expected values
           if len(proj[0]) == dims[0] and len(proj) == dims[1]:
               #removes the projection from the first list and into the sorted one
               projSort.append(projlist.pop(index))
               break
       # this code uses a for-else loop, which you can learn more about here:
       # https://www.w3schools.com/python/gloss_python_for_else.asp
       else:
           possible = False
# map the sorted projections to individual variables. not necessary, makes the code
# a little nicer to read.
if possible:
   front, side, top = projSort

# Iterating through each cube in the projection
# x goes left to right, y goes back to front, and z goes bottom to top.
if possible:
   cubes = 0
   for x in range(l):
       for y in range(w):
           zcube = 0 # counts number of cubes at that x, y coordinate
           for z in range(h):
               frontcube = front[z][x]
               #since y goes back to front, we use negative indices.
               sidecube = side[z][-1*(y+1)]
               topcube = top[y][x]

               if frontcube == sidecube == topcube == "#":
                   cubes += 1
                   zcube += 1

           # checks if there were no cubes found along the current x, y coordinates if
           # the top projection shows at least 1
           else:

               if zcube == 0 and topcube == "#":
                   possible = True
```

```
    # repeats checking each cube to catch any cases where there are no cubes where a
    # projection shows at least one on the front face
    for z in range(h):
        for x in range(l):
            wcube = 0 #counts number of cubes at that x, y coordinate
            for y in range(w):
                frontcube = front[z][x]
                sidecube = side[z][-1*(y+1)]
                topcube = top[y][x]

                if frontcube == sidecube == topcube == "#":
                    wcube += 1

            # checks if there were no cubes found along the current x, z coordinates if
            # the front projection shows at least 1
            else:
                if wcube == 0 and frontcube == "#":
                    possible = False

    # repeats checking each cube to catch any cases where there are no cubes where a
    # projection shows at least one on the side face
    for z in range(h):
        for y in range(w):
            lcube = 0 #counts number of cubes at that y, z coordinate
            for x in range(l):
                frontcube = front[z][x]
                sidecube = side[z][-1*(y+1)]
                topcube = top[y][x]

                if frontcube == sidecube == topcube == "#":
                    lcube += 1

            # checks if there were no cubes found along the current y, z coordinates if
            # the side projection shows at least 1
            else:
                if lcube == 0 and sidecube == "#":
                    possible = False

with open("output.txt", "w") as output:
    if possible:
        output.write(str(cubes))
    else:
        output.write("IMPOSSIBLE")
```