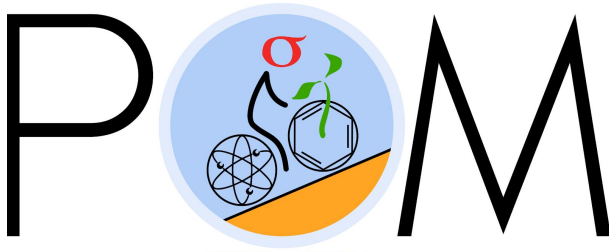
PROBLEM OF THE MONTH



October, 2020

MATHEMATICS

5 points:

Alice and Bob have independently proofread the same manuscript. Alice, the more experienced editor, has found 600 misprints while Bob managed to find only 400. When they compared their proofs they discovered that 300 of those misprints had been found by both of them. Estimate how many misprints in the manuscript are **not** found by either Bob or Alice. Please try to explain the assumptions you used in your estimate.

Hint: Assume that there are N misprints in the manuscript. Compute the probability for Alice to find a misprint and similarly for Bob. What is the probability that a given mistake is found by both of them?

Answer: 100

Solution: Let us denote the total number of misprints in the manuscript N. Then Alice found the fraction $\frac{600}{N}$ and Bob found the fraction $\frac{400}{N}$. Let us assume that one can neglect statistical fluctuations and those fractions represent the probabilities of finding mistakes by Alice and Bob. We expect then that the probability of the same mistake to be found by both Alice and Bob is $\frac{600}{N}\frac{400}{N} = \frac{300}{N}$. We find the total number of misprints $N = \frac{600*400}{300} = 800$. As the total number of misprints found by Alice and Bob together is $600 + 400 - 300 = 700,$ we expect about 100 misprints still left in the manuscript. As we neglected statistical fluctuations this answer is only an estimate, not the exact number.

10 points: Humpty and Dumpty found a treasure chest with 140 coins. They decided to split it by playing the following game. They toss a fair coin and write down the results “Heads (H)” and “Tails (T)”. If they have three heads in a row the game is stopped and Humpty wins. If they get three tails in a row before heads, Dumpty is declared the winner. The winner of the game

takes all 140 coins. Unfortunately, the game was interrupted by all the king's men when they had the following sequence of tosses: HTTHTHTT. Help Humpty and Dumpty split the coins. Dumpty claims that he had a bigger chance of winning and should take all the coins. Humpty claims that he also had a chance to win and should get his share. How many coins should be given to **Humpty** to take into account his chances to win in case the game would not be interrupted?

Hint: What would be the probability for Humpty to win the game if they continued to play? Try to relate such probabilities for sequences ending with HTT, HT, TH, THH.

Answer: 40

Solution:

Let us assume that a sequence of tosses ends with HTT. The probability P_{HTT} for Humpty to win can be found as

$P_{HTT} = \frac{1}{2} \times 0 + \frac{1}{2} \times P_{TH}$.

Indeed, with the probability ½ the next toss is T and Humpty loses and with the probability ½ the next toss is H and the probability for Humpty to win at that point becomes P_{TH}. By similar arguments we find

$P_{HT} = \frac{1}{2} \times P_{HTT} + \frac{1}{2} \times P_{TH}$,

$P_{TH} = \frac{1}{2} \times P_{HT} + \frac{1}{2} \times P_{THH}$,

$P_{THH} = \frac{1}{2} \times P_{HT} + \frac{1}{2} \times 1$.

Solving this system of equations we obtain $P_{HTT} = \frac{2}{7}, P_{HT} = \frac{3}{7}, P_{TH} = \frac{4}{7}, P_{THH} = \frac{5}{7}$. In particular the probability of Humpty winning at the sequence given in the problem is $P_{HTT} = \frac{2}{7}$ (and for Dumpty $1 - \frac{2}{7} = \frac{5}{7}$). We conclude that Humpty should get $\frac{2}{7} \times 140 = 40$ coins.

PHYSICS

5 points: Sila and Numan are a pair of asteroids that orbit our Sun. In addition, they orbit each other, making one full turn around their common center of mass every 12.5 days. The distance between them is 2800 km. Find the total mass of both asteroids from these data. Gravitational Constant is G=6.67 × 10^{-11} N kg^{-2} m^2.

Hint: Let R be distance between the asteroids, and let their masses be m_1 and m_2. How far is each of them from their center of mass (which is the center of each ordit)?

Answer: $1.1 \cdot 10^{19}$ kg

Solution: The asteroids follow circular orbits around their common center of mass. The gravity force, F, acting on each of the asteroids, renders their centripetal acceleration. According to Newton's laws,

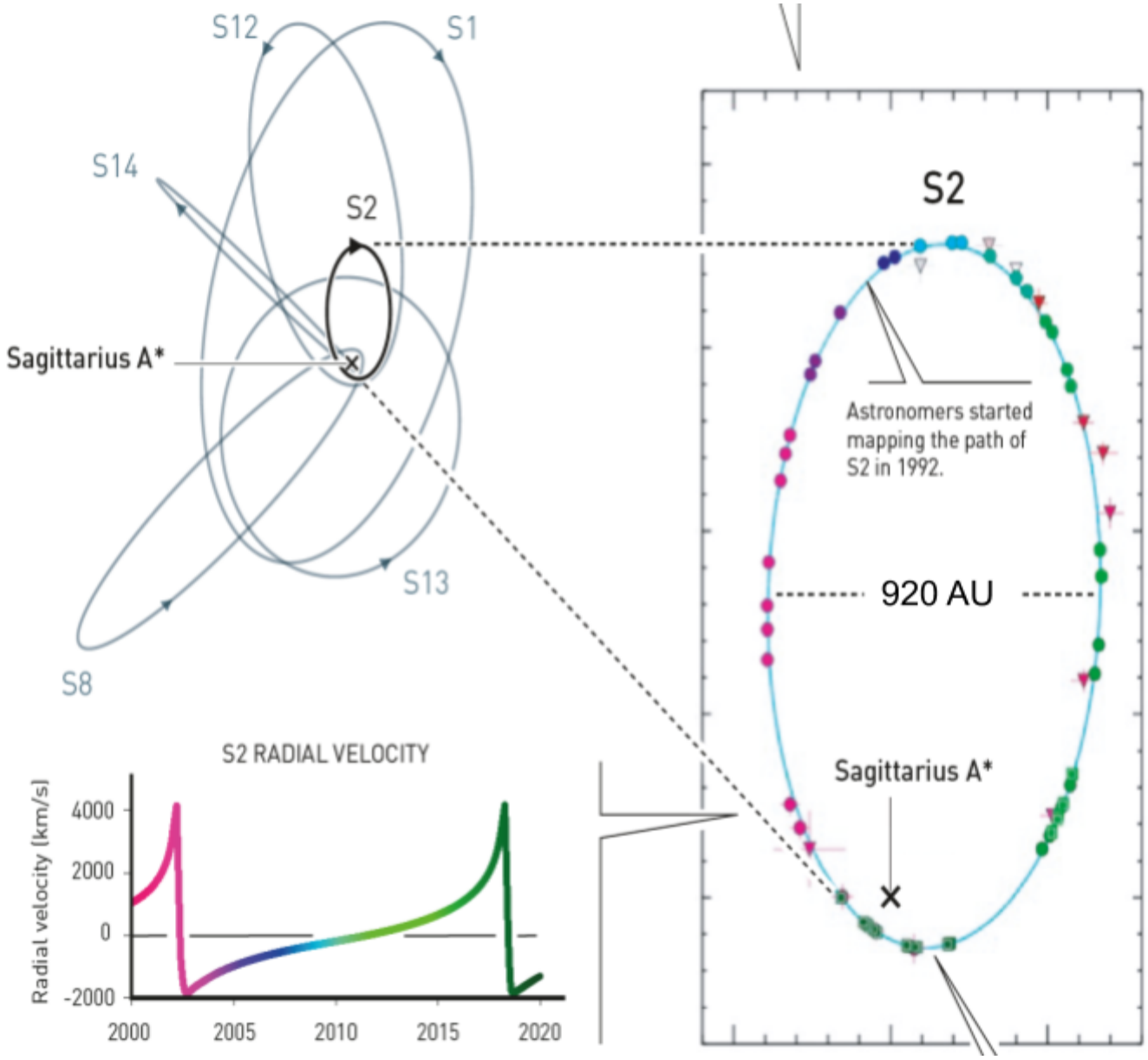
$$F = G\frac{m_1 m_2}{(r_1+r_2)^2} = m_1\omega^2 r_1 = m_2\omega^2 r_2 ,$$

where m_1, r_1 and m_2, r_2 are the mass and the radius of orbit of Sila and Numan, respectively, and $\omega = \frac{2\pi}{T}$ is the common angular velocity of their rotation corresponding to the period of

$$T = 12.5\ days = 12.5 \cdot 24 \cdot 3600 = 1.08 \cdot 10^6 s .$$

From the Newton's laws we find, $m_2 = \frac{\omega^2 r_1 (r_1+r_2)^2}{G}$ and $m_1 = \frac{\omega^2 r_2 (r_1+r_2)^2}{G}$. Hence,

$$m_1 + m_2 = \frac{\omega^2 (r_1+r_2)^3}{G} = \left(\frac{2\pi}{T}\right)^2 \frac{(r_1+r_2)^3}{G} = \left(\frac{2 \cdot 3.14}{1.08 \cdot 10^6}\right)^2 \frac{\left(2.8 \cdot 10^6\right)^3}{6.67 \cdot 10^{-11}} = 1.1 \cdot 10^{19} kg.$$



10 points: 2020 Nobel prize in Physics was awarded (in part) for the discovery of a supermassive black hole Sagittarius A* in the center of our galaxy, Milky Way. Observations of several stars orbiting it, particularly star S2, have been used to determine the mass of this object. Use the figure above and information given on that figure to estimate that mass in terms of the mass of our Sun. Note that one astronomical unit (AU) is equal to the radius of Earth's orbit (when it is approximated by a circle). Assume that S2 orbit in the Figure is drawn to scale.

Hint: For a circular orbit, find how the period depends on the mass of the star (or a black hole) in the center, and on radius. Based on Kepler's third law, which parameter of the elliptical orbit determines the period?

Answer: about $4 \cdot 10^6$

Solution: According to Kepler's third law, the period ***T*** depends on the major semi-axis ***a*** of the orbit , which is 1 a.u. for Earth orbiting the Sun, and approximately 1000 a.u. for star S2 orbiting the Sagittarius A*. Period of the orbital motion is 1 year for Earth, and approximately 16 years for S2. By considering an object of mass ***m*** on a circular orbit of radius ***a*** around a much heavier mass ***M,*** we find the following relationship between period ***T*** of the orbital motion, ***a*** and ***M***:

$$\frac{GMm}{a^2} = \frac{mv^2}{a} = \frac{4\pi^2 a^2 m}{aT^2}$$

Hence,

$$M = const \times \frac{a^3}{T^2}$$

This gives the mass of Sagittarius A* in terms of mass of our Sun: $M \approx \frac{1000^3}{16^2} \approx 4 \cdot 10^6$.

CHEMISTRY

This month, the topic is: **Electrolytic dissociation**.
IMPORTANT! In this PoM season, we do an experiment: each month, an online lecture will be given. This lecture may be helpful for those who want to solve Chemistry PoMs, although it is not supposed to provide direct hints.
This month, the lecture will be on Oct 18 morning. At 10:30, a Zoom conference will start where September PoM solutions will be discussed. After that, approximately at 11:00, the lecture starts.
To join the Zoom conference, use this link:
https://us02web.zoom.us/j/4817690592?pwd=T2djSjRETEpDSHFZdWJpYlBTYzdjQT09
Meeting ID: 481 769 0592
Passcode: 879615
If you are unable to connect, email to mark.lukin@gmail.com

5 points:

This problem describes a real case that took place in one research laboratory at Stony Brook University. A researcher was preparing artificial seawater to maintain the culture of marine algae using that can be found here: http://cshprotocols.cshlp.org/content/2012/2/pdb.rec068270.full , and she found that there is no magnesium sulfate in the lab. That was a serious problem, because the seawater was needed urgently, otherwise the algal culture would die.
The researcher decided to modify the recipe and to make the same artificial water using a somewhat different set of chemicals. That worked fine, and the algal culture was saved.
What other common chemicals did the researcher use to prepare seawater of exactly the same composition? What is the amount of each component in grams per one liter of the solution?

Hint:

All components of the artificial seawater are electrolytes. That is a hint.

Solution:

There are no “salt molecules” in aqueous solution, because NaCl, KCl, $CaCl_2$, $MgCl_2$, and $MgSO_4$ dissociate when dissolved in water. That means, there are only Na+, K+, Ca^{2+}, Mg^{2+}, Cl^-, and SO_4^{2-} ions in artificial seawater, and their concentrations are 450 mM, 10 mM, 9 mM, 46 mM (30 + 16), 538 mM (450 +10 + 2×9 + 2×30), and 16 mM, accordingly. Therefore, this solution can be prepared by using any combination of salts that contains that amount of each of these ions. Thus, instead of taking 30 mM of magnesium chloride ($MgCl_2$) and 16 mM of magnesium sulfate ($MgSO_4$), one can take 46 mM of magnesium chloride and 16 mM of sodium sulfate. This combination of salts yields needed concentration of Mg^{2+} (46 mM) and SO_4^{2-} (16 mM), but the amount of Cl- increases by 32 mM (we have extra 16 mM of $MgCl_2$, upon dissociation, it produces 32 mM of chloride ions), and the amount of Na+ also increases by 32

mM (16 mM of sodium sulfate produce 32 mM of Na^+ upon dissolution). To compensate for that change, we have to decrease the amount of NaCl by 32 mM.
Other combinations of salts may also be used, so this problem has multiple solutions.

10 points:

Thymol blue and methyl orange are indicator dyes that change color in the presence of acids. You have three bottles with 0.1M solutions of HCl, HF, and acetic acid, and 0.01M solution of phosphoric acid. The bottles have no labels. Is it possible to figure out which is in each bottle using there two indicator solutions? Which acids are possible to discriminate, and which are not? Why?

Hint:

pH of the solution of an acid can be calculated from its dissociation constant and concentration.

Solution:

Thymol blue and methyl orange change color at different pH (for the former, the transition point is below 2, for the latter, it is in between nearly 3). That means, it may be possible to discriminate the solutions of these acids if their pH are different. Let's check if that is possible. pH is a parameter that describes concentration of hydrogen ions in solution. If concentration of H^+ is, for example, 10^{-3}, pH of this solution is 3. That means we need to calculate concentration of H^+ in all three solutions (HCl, HF, and acetic acid), and that will give us their pH.

For a dissociation reaction:

$$HX \rightleftharpoons H^+ + X^-$$

dissociation constant is defined as:

$$K_d = [H^+][X^-]/[HX] \qquad (1)$$

where a symbol in square brackets (e.g. [A]) denotes actual concentration of the species A. In that sense, it is different from the nominal concentration (which we denote as C): if we prepared, e.g., 1 mM solution of HF, its nominal concentration (C_{HF}) is 1 mM, but we don't know the actual concentration, because some fraction of HF molecules dissociated, so we denote the concentration of HF molecules, i.e., the molecules that had not dissociated, as [HF]. Accordingly, the dissociated molecules yield the ions H^+ and F^-. We do not know their concentrations either, so we denote them as $[H^+]$ and $[F^-]$.

We need to calculate $[H^+]$. We do not know $[F^-]$ and [HF], but we know K_{HF} and C_{HF}.

Since the equation (1) has three unknown variables, we cannot calculate $[H^+]$ from it. However, using simple logic, it is possible to simplify that equation. Indeed, these three variables are not independent: First, any dissociation event leads to formation of one cation and one anion, which means that:

$$[H^+] = [F^-] \quad (2)$$

Second, in any dissociation event that leads to formation of one cation (or anion), one molecule of acid disappears, so the sum of [HF] and $[H^+]$ (or $[F^-]$) always equal to the nominal concentration of HF (C_{HF}). That means:

$$[HF] = C_{HF} - [H^+] \quad (3)$$

That means we can express two variables in the equation (1) through another one. We are interested to know $[H^+]$, so let's get rid of other two. Let's consider a general case, so instead of HF we speak about some generic acid HX.

Step 1:

$K_d = [H^+][X^-]/[HX] = [H^+][H^+]/[HX] = [H^+]^2/[HX]$ (because of (2))

Step 2:

$K_d = [H^+]^2/[HX] = [H^+]^2/(C_{HX} - [H^+])$ (because of (3))

The equation:

$$K_d = [H^+]^2/(C_{HX} - [H^+]) \quad (4)$$

is a quadratic equation with one variable ($[H^+]$), so it can be solved, and $[H^+]$ calculated. However, that is needed only when $C_{HX} \approx [H^+]$. For strong acids, $C_{HX} << [H^+]$ (because the acid is essentially dissociated). For weak acids, the situation is opposite: only a small fraction of a weak acid dissociate, so in other words, $C_{HX} >> [H^+]$. That makes our life much easier, because we don't need to solve any quadratic equation.

Case 1. Strong acids, e.g. HCl and/or dilute solutions.
In that case, we need to transform the equation (4) into a standard quadratic equation:

$$K_d = [H^+]^2/(C_{HX} - [H^+]) \rightarrow [H^+]^2 = K_dC_{HX} - K_d[H^+] \rightarrow [H^+]^2 + K_d[H^+] - K_dC_{HX} = 0 \quad (5)$$

but we do not need to solve it, because we remember that $[H^+]$ is small, which means that $[H^+]^2$ is almost negligible, so we can assume it is equal to zero, so the equation transforms to:

$$- K_dC_{HX} + K_d[H^+] = 0,$$

or

$[H^+] = C_{HX}$

In other words, if **HCl concentration is 0.1 M, $[H^+]$ is approximately 0.1 M, so pH is 1**. (HCl is a very strong acid).
Similarly,
In 0.001 M phosphoric acid solution, which is a very dilute solution [H+] is clove to the nominal concentration of the acid, i.e. 10^{-3},
and pH is 3.

Case 2. Weak acids.

In weak acids, only a small fraction of HX dissociate ($C_{HX} >> [H^+]$), so it would be reasonable to assume that the denominator in the equation (4) is approximately equal to C_{HF} (for $C_{HF} >> [H+]$), and the equation (4) transforms to:

$$K_d = [H^+]^2/C_{HF} \quad (6)$$

Equation (5) immediately gives $[H^+]$:

$$[H+] = \sqrt{(K_d C_{HF})} \quad (7)$$

So for a weak acid, $[H^+]$ is a geometric mean of its concentration and dissociation constant.

All of that allows us to calculate pH of HCl, HF, acetic and phosphoric acids. HCl is a strong acid ($K_d = 10^6$). HF, acetic acid, and phosphoric acids are weak to moderately weak (K_ds are approximately 10^{-3}, 10^{-5}, and 10^{-2}, accordingly). Therefore, we can use the formula (7), so $[H^+]$ in these solutions are:

In 0.1 M HF solution, [H+] is $\sqrt{(10^{-1}\times10^{-3})} = 10^{-2}$,
and pH is 2.
In 0.1 M acetic acid solution, [H+] is $\sqrt{(10^{-1}\times10^{-5})} = 10^{-3}$,
and pH is 3.
In summary, we have two solutions with pH of 3 (acetic and phosphoric acids), one solution with pH of 2 (HF), and one solution with pH of 1 (HCl). pH HCl is below the transition point of both indicators, pH of HF is below the transition point of methyl orange and close to the transition point of bromophenol blue, and pH of other two solutions is above the transition point of both indicators. That means it is possible to identify HCl and HF, but it would not be possible to tell which solution is acetic acid, and which solution is phosphoric acid.

BIOLOGY

5 points:

Genetically modified crops have been developed that produce a protein that makes the plant resistant to insect pests. Other genetic modification makes the crops more resistant to chemicals that kill plants (herbicides). Describe three potential biological risks of large-scale cultivation and use of such genetically modified plants. Describe three potential benefits of large-scale cultivation of such a plant (this may include potential for the biosphere, for humans, or for some specific biological species).

Answer:

Possible risks:
Unknown human/other animals health risk due to consuming GM proteins; Developed resistance in pest species; Spread of genetic modification to non_GM plants; GM plants out of complete native species; Loss of biodiversity; Use of herbicide harms non-target species; Invasive disease wiping out the monoculture; Disruption within food chain.

Benefits:

Increased productivity saves wild nature, because no additional land is needed for agriculture; Increased resistance to insect pests decrease the amount of herbicides used for plant's cultivation; the use of herbicide-tolerant plants avoids weeding the fields, thus saving fuel and machinery, thereby decreasing carbon dioxide emission.

10 points:

In the case of COVID-19, public health experts at the CDC often make predictions about how different public policy decisions might influence outcomes over the next weeks or months. How do they do this? In epidemiology, mathematical models are often used to probe how different factors ("parameters") affect viral transmission, over an entire population, over time. The power of "models" is that, by running simulations, we can see the impact of multiple hypothetical scenarios--essentially running experiments "in silica."

Using this link, you can access a GUI and visualization of one viral model. The parameters of this model, for some particular virus, include: (1) the density of the population, (2) the population turnover (i.e., death & reproduction), (3) the degree of immunity, the degree of infectiousness (or transmissibility), and (4) the duration of infectiousness.

Assuming a closed community of 100 individuals, with a highly-infectious virus (80%), and an infectiousness duration of one year:

a) What happens to the total population if the virus is completely lethal (Virus X)?
b) What happens if the virus is only 50% lethal (Virus Y)?
c) In the 30-50% range (Virus Z), what kind of behavior do you see in terms of the viral population? Why.
d) As public health officer, with limited resources and with the sole perspective of saving the maximum number of lives, would it be a more effective one year strategy to reduce viral infectiousness by 20% (e.g., by promoting behaviors like social distancing and mask use) or by reducing viral mortality by 20% (e.g., by developing more effective medical treatments). Use the model to provide evidence for your public health policy. Would you make the same recommendation for both Viruses X and Y?

Solution:

a) What happens to the total population if the virus is completely lethal (Virus X)?

If the chance of recovery is 0%, the population initially plummets, but then the virus eventually dies as well (by about 176 weeks, or 3.38 years), leaving the population completely healthy.

b) What happens if the virus is only 50% lethal (Virus Y)?

With 50% lethality, the virus is able to sustain itself indefinitely (>25 years).

c) In the 30-50% range (Virus Z), what kind of behavior do you see in terms of the viral population? Why?

The number of sick people (and thus total amount of virus) oscillates, because it's close to the "critical point" where there is divergence between two trajectories. When there are too many deaths, it then reduces the viral population, which in turn reduces the number of deaths, which then increases the viral population (i.e.,, Lotka-Volterra).

d) As public health officer, with limited resources and with the sole perspective of saving the maximum number of lives, would it be a more effective one year strategy to reduce viral infectiousness by 20% (e.g., by promoting behaviors like social distancing and mask use) or by reducing viral mortality by 20% (e.g., by developing more effective medical treatments). Use the model to provide evidence for your public health policy. Would you make the same recommendation for both Viruses X and Y?

As you can see from the modelling results, the outcome of simulations strongly depends on initial conditions, so in each case the answer is possible only after the modelling is performed using some specific set of parameters describing infectiousness and lethality. At some starting conditions, even a small decrease of infectiousness may lead to full elimination of the disease from the population, but an accurate answer is possible only after modelling.

LINGUISTICS

5 points:

Consider the translations of the following phrases from an Indo-Iranian language:

1. *dusti hubi hamsoyai shumo* – “a good friend of your neighbor”
2. *hamsoyai dusti hubi shumo* – “a neighbor of your good friend”
3. *hamsoyai hubi dusti shumo* – “a good neighbor of your friend”

Determine the translations of individual words and explain your reasoning.

Hint:

Answer:

Hubi = good
Shumo = your
Dusti = friend
Hamsoiai = neighbor

Solution:

Let’s first compare 1 and 3. In the translations, *friend* and *neighbor* are exchanged. In the sentences, *dusti* and *hamsoyai* are exchanged. Therefore, these two words correspond to *friend* and *neighbor* (in some order).
Now let’s compare 2 and 3. The only difference in translation is which word an adjective *good* modifies. In the sentences, *dusti* and *hubi* were exchanged. We know that dusti is a noun, therefore, *hubi* means *good*, and *shumo* means *your*.
Now we have to check which noun means what. If *dusti* means *friend* and *hamsoiai* means *neighbor*, then we have a reasonable structure: adjectives come after nouns, possessives come after adjectives. If these words are translated in a different order, then 1 and 2 will have to be generated by inconsistent rules.

10 points:

Below you’re given the words in one of the ancient alphabets and their pronunciations.

ᚹᚴᚱᛑ	yagɨ	‘enemy’
ᚴᛇᚠᚺ	kɨlɨn	‘to make’
ᛇᛒᛉᚠᛟ	bilge	‘wise’
ᚺᛞᛝ	adak	‘leg’
ᚠᚲ	eki	‘two’
ᛇᛋᚺ	kara	‘black’
ᛇᛚᛇᛚ	balbal	‘statue’
ᚾᛒᚺᛇ	ingek	‘cow’
ᚸᚱᛜ	tag	‘mountain’
ᛉᛏᚹ	yer	‘earth’
ᛝᛇᚠ	ɨd	‘to send’
ᛉᛋᛚ	bars	‘tiger’
ᛒᛁᚠ	isig	‘labor’
ᚠᚳᛉᚲ	kelti	‘they came’
ᛉᚠᚠ	esir	‘defeated’

a) Please write the following words:

elt	‘lead’
bark	‘house’
kan	‘blood’
ab	‘hunting’
altɨ	‘six’
bes	‘five’
birle	‘together’
kalɨn	‘fat’
bilig	‘title’
yablak	‘bad’

b) Figure out in how many ways the following word can be read and list them all.

Answer:

elt	‘lead’	ᚻᛉ	bes	‘five’	ᛁᛟ
bark	‘house’	ᚸᚺᚤ	birle	‘together’	ᛇᛉᛏᛚᛟ
kan	‘blood’	ᚲᚸ	kaḷin	‘fat’	ᚲᛚᛇᚸ
ab	‘hunting’	ᚤ	bilig	‘title’	ᛒᛉᛚᛟ
altḭ	‘six’	ᛚᛃᛇ	yablak	‘bad’	ᚸᛇᚤᛞ

ᛃᚲᚴᛚᛊ can be read as (a)sɨg(a/ɨ/-)t(ɨ), where parenthesized vowels are optional.

Solution:

First, comparing say *balbal* and *bars*, we determine that the order is from right to left. Further, we can notice that words can be divided into two groups, one group has vowels *a* and *ɨ*, the other -- vowels *e* and *i*. It seems like *a* and *e* are denoted as ᛇ, and *i* and *ɨ* are denoted as ᛚ. Also, *a/e* are only indicated at the end of the word, while *i/ɨ* are represented at any position, but only once -- where it is encountered first. We can also see that soft consonants (next to i and e) and hard consonants (next to ɨ and a) are spelled differently.

Here is the table of all characters (where apostrophe signifies a soft consonant, occurring in words with [e] and [i]):

ᛇ [a]	ᛇ [e]
ᛚ [ɨ]	ᛚ [i]
ᚤ [b]	ᛒ [b’]
ᛝ [d]	Not in the data! [d’]
ᚴ [g]	ᛄ [g’]
ᛞ [y]	ᛩ [y’]
ᚸ [k]	ᚳ [k’]
ᛎ [l]	ᛉ [l’]
ᚲ [n]	ᚻ [n’]
ᚺ [r]	ᛥ [r’]
ᛊ [s]	ᛁ [s’]
ᛃ [t]	ᚻ [t’]

Hence, the spellings are:

elt	‘lead’	ᚼᛉ	bes	‘five’	ᛁᛟ
bark	‘house’	ᚮᚺᚤ	birle	‘together’	ᛦᛉᛏᛚᛟ
kan	‘blood’	ᚲᚮ	kalɨn	‘fat’	ᚲᛚᛇᚮ
ab	‘hunting’	ᚤ	bilig	‘title’	ᛓᛉᛚᛟ
altɨ	‘six’	ᛚᛜᛇ	yablak	‘bad’	ᚮᛇᚤᛞ

Further, the word ᛜᚸᛚᛊ can be read according to the following pattern: (a)sɨg(a/ɨ/-)t(ɨ), where parenthesized vowels are optional.

COMPUTER SCIENCE

- Your program should be written in Java or Python-3
- No GUI should be used in your program: eg., easy gui in Python
- All the input and output should be via files with specified in the problem names
- 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!

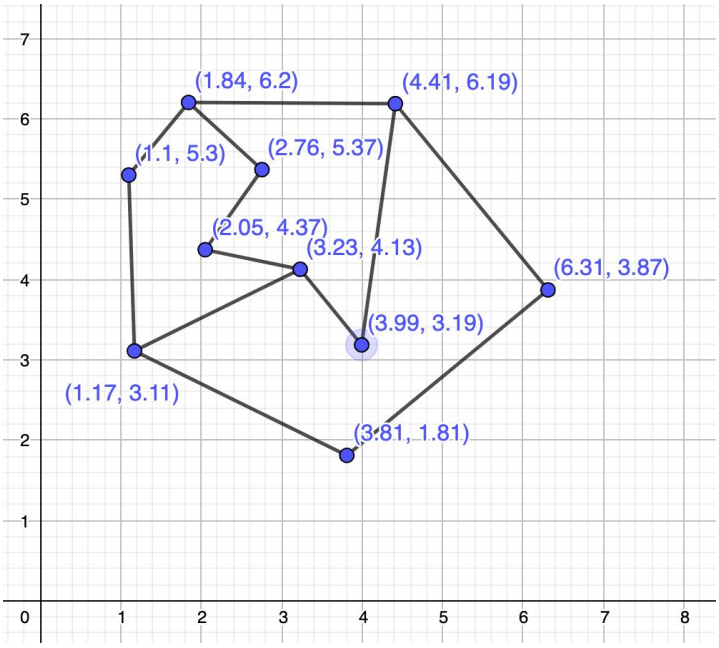
Introduction:

To maintain proper law and order in SigmaCity, Sigma police force is split into a set of precincts. Each police precinct serves an area in the shape of a polygon, which is defined as a set of (x,y) coordinates of the polygon's vertices (x and y are floating point numbers). Precinct areas do not overlap and completely cover the whole SigmaCity, which also has the shape of a polygon. Each precinct is assigned a unique number (positive integer).

5 points:

Your program should find the area of the largest police precinct in SigmaCity.

The program should read the input file **input.txt**. First line contains an integer number of police precincts in Sigma City. It is followed by that number of rows, one row per Sigma police precinct. Each row contains a series of space-separated values, starting with the precinct number and followed by pairs of (x,y) floating point coordinates of vertices defining the precinct area. Example input file:




```
3
1 1.17 3.11 1.1 5.3 1.84 6.2 2.76 5.37 2.05 4.37 3.23 4.13
2 1.84 6.2 4.41 6.19 3.99 3.19 3.23 4.13 2.05 4.37 2.76 5.37
3 3.99 3.19 4.41 6.19 6.31 3.87 3.81 1.81 1.17 3.11 3.23 4.13
```

Your program should produce the output file **output.txt**, which consists of a single floating point number representing the area of the largest precinct.

Solution:

Java:

```
// using the formula from https://en.wikipedia.org/wiki/Polygon for non-self-intersecting
// polygons

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.*;
import static java.lang.Math.abs;

public class Precinct5 {
  private int n;
  private Map<Integer, List<List<Double>>> precincts = new HashMap<>(); // precinct number ->
                                                                        // list of coordinates
  private double aMax = 0;

  void input() throws Exception {
    BufferedReader reader = new BufferedReader(new FileReader("input.txt"));
    String line = reader.readLine().trim();
    String[] nums = line.split("\\s*[\\s,]\\s*");
    // the following will croak if the elements are not all integers
    int[] numbers = Arrays.stream(nums).map(s ->
Integer.valueOf(s)).mapToInt(Integer::intValue).toArray();
    if(numbers.length != 1)
      throw new Exception("1st line must have 1 integer");
    n = numbers[0];

    for(int i=0; i<n; i++) {
      line = reader.readLine().trim();
      nums = line.split("\\s*[\\s,]\\s*");
      double[] ns = Arrays.stream(nums).map(s ->
Double.valueOf(s)).mapToDouble(Double::doubleValue).toArray();
      List<List<Double>> a = new ArrayList<>();
      for(int j=1; j<ns.length; j+=2) {
        List<Double> e = new ArrayList<>();
        e.add(ns[j]);
        e.add(ns[j+1]);
        a.add(e);
      }
      precincts.put((int)ns[0], a);
    }
  }

  void calc() {
    for(List<List<Double>> cs : precincts.values()) {
      double a = 0;
      for(int i=0; i<cs.size(); i++) {
        double xi = cs.get(i).get(0);
        double yi1;
        if(i+1 < n)
          yi1 = cs.get(i+1).get(1);
        else
          yi1 = cs.get(0).get(1);
```

```
        double xi1;
        if(i+1 < n)
          xi1 = cs.get(i+1).get(0);
        else
          xi1 = cs.get(0).get(0);
        double yi = cs.get(i).get(1);
        a += xi*yi1 - xi1*yi;
      }
      a /= 2.0;
      a = abs(a);
      if(a > aMax)
        aMax = a;
    }
  }

  void output() throws Exception {
    try(FileWriter out = new FileWriter("output.txt")) {
      out.write(String.format("%f\n", aMax));
    }
  }

  public static void main(String[] args) throws Exception {
    Precinct5 p = new Precinct5();
    p.input();
    p.calc();
    p.output();
    System.out.println("end.");
  }
}
```

Python:

```
# using the formula from https://en.wikipedia.org/wiki/Polygon for non-self-intersecting
polygons

import re

# read and parse input file
precincts = {} # precinct number -> list of coordinates
with open("input.txt") as in_file:
  line = in_file.readline()
  numbers_str = re.split(r"[\s,]\s*", line.strip()) # split by either white spaces or commas
  # the following will croak if the elements are not all integers
  numbers = [int(x) for x in numbers_str]
  if len(numbers) != 1:
    raise Exception("first line must have 1 number")
  n = numbers[0]

  for i in range(n):
    line = in_file.readline()
    numbers_str = re.split(r"[\s,]\s*", line.strip()) # split by either white spaces or commas
    # the following will croak if the elements are not all floats
    numbers = [float(x) for x in numbers_str]
    precincts[int(numbers[0])] = [(numbers[i], numbers[i+1]) for i in range(1,len(numbers),2)]

a_max = 0
for p, cs in precincts.items():
  a = 0
```

```
  for i in range(len(cs)):
    xi  = cs[i][0]
    yi1 = cs[i+1][1] if i+1<n else cs[0][1]
    xi1 = cs[i+1][0] if i+1<n else cs[0][0]
    yi  = cs[i][1]
    a += xi*yi1 - xi1*yi
  a /= 2
  a = abs(a)
  if a > a_max:
    a_max = a

with open("output.txt", "w") as out_file:
  out_file.writelines(f"{a_max}\n")

print("end.")
```

10 points:

Your program will be finding which police precinct covers a particular location in Sigma City given its coordinates.

The program should read the input file **input.txt**. First line contains an integer number of police precincts in Sigma City. It is followed by that number of rows, one row per Sigma police precinct. Each row contains a series of space-separated values, starting with the precinct number and followed by pairs of (x,y) floating point coordinates of vertices defining the precinct area. Finally, the last row contains x, y coordinates of the given location in Sigma City.
Example input file:

```
3
1 1.17 3.11 1.1 5.3 1.84 6.2 2.76 5.37 2.05 4.37 3.23 4.13
2 1.84 6.2 4.41 6.19 3.99 3.19 3.23 4.13 2.05 4.37 2.76 5.37
3 3.99 3.19 4.41 6.19 6.31 3.87 3.81 1.81 1.17 3.11 3.23 4.13
4.12 4.55
```

Your program should produce the output file **output.txt**, which consists of a single integer number representing the number of the precinct covering the given location or 0 if the given coordinates lay outside of Sigma City (i.e. outside of all its police precincts).

Solution:

Java:

```
// the explanations can be found here:
http://www.dcs.gla.ac.uk/~pat/52233/slides/Geometry1x1.pdf
// and here:
https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/
// and here: https://en.wikipedia.org/wiki/Point_in_polygon
```

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.*;

public class Precinct10 {
  static int INF = 10000;

  static class Point {
    double x;
    double y;
    public Point(double x, double y) {
      this.x = x;
      this.y = y;
    }
  };

  private int n;
  private Map<Integer, Point[]> precincts = new HashMap<>(); // precinct number -> polygon
  private Point point;

  void input(String fname) throws Exception {
    BufferedReader reader = new BufferedReader(new FileReader(fname));
    String line = reader.readLine().trim();
    String[] nums = line.split("\\s*[\\s,]\\s*");
    // the following will croak if the elements are not all integers
    int[] numbers = Arrays.stream(nums).map(s ->
Integer.valueOf(s)).mapToInt(Integer::intValue).toArray();
    if(numbers.length != 1)
      throw new Exception("1st line must have 1 integer");
    n = numbers[0];

    for(int i=0; i<n; i++) {
      line = reader.readLine().trim();
      nums = line.split("\\s*[\\s,]\\s*");
      double[] ns = Arrays.stream(nums).map(s ->
Double.valueOf(s)).mapToDouble(Double::doubleValue).toArray();
      if(ns.length%2 == 0)
        throw new Exception("there must be a precinct number followed by even number of
coordinates");
      Point[] polygon = new Point[(ns.length-1) / 2];
      for(int j=1; j<ns.length; j+=2) {
        polygon[(j-1)/2] = new Point(ns[j], ns[j+1]);
      }
      precincts.put((int)ns[0], polygon);
    }

    line = reader.readLine().trim();
    nums = line.split("\\s*[\\s,]\\s*");
    double[] ns = Arrays.stream(nums).map(s ->
Double.valueOf(s)).mapToDouble(Double::doubleValue).toArray();
    if(ns.length != 2)
      throw new Exception("last line must have 2 numbers");
    point = new Point(ns[0], ns[1]);
  }

  void output(String fname, int precinct) throws Exception {
```

```
    try(FileWriter out = new FileWriter(fname)) {
      out.write(String.format("%d\n", precinct));
    }
  }

  // given 3 collinear points p, q, r, the function checks if point q lies on line segment
'pr'
  static boolean onSegment(Point p, Point q, Point r) {
    if(q.x <= Math.max(p.x, r.x) && q.x >= Math.min(p.x, r.x) &&
       q.y <= Math.max(p.y, r.y) && q.y >= Math.min(p.y, r.y)) {
      return true;
    }
    return false;
  }

  // find orientation of ordered triplet (p, q, r)
  // the function returns following values
  //   0 --> p, q and r are collinear
  //   1 --> clockwise
  //   2 --> counterclockwise
  static int orientation(Point p, Point q, Point r) {
    double val = (q.y - p.y) * (r.x - q.x) - (q.x - p.x) * (r.y - q.y);
    if(val == 0) // accuracy problem
      return 0; // collinear
    return (val > 0) ? 1 : 2; // clock or counterclock -wise
  }

  // the function that returns true if line segment 'p1q1' and 'p2q2' intersect
  static boolean doIntersect(Point p1, Point q1, Point p2, Point q2) {
    // find the four orientations needed for general and special cases
    int o1 = orientation(p1, q1, p2);
    int o2 = orientation(p1, q1, q2);
    int o3 = orientation(p2, q2, p1);
    int o4 = orientation(p2, q2, q1);

    // general case
    if(o1 != o2 && o3 != o4)
      return true;
    // special cases
    // p1, q1 and p2 are collinear and p2 lies on segment p1q1
    if(o1 == 0 && onSegment(p1, p2, q1))
      return true;
    // p1, q1 and p2 are collinear and q2 lies on segment p1q1
    if(o2 == 0 && onSegment(p1, q2, q1))
      return true;
    // p2, q2 and p1 are collinear and p1 lies on segment p2q2
    if(o3 == 0 && onSegment(p2, p1, q2))
      return true;
    // p2, q2 and q1 are collinear and q1 lies on segment p2q2
    if(o4 == 0 && onSegment(p2, q1, q2))
      return true;
    // doesn't fall in any of the above cases
    return false;
  }

  // returns true if the point p lies inside the polygon[] with n vertices
  static boolean isInside(Point[] polygon, int n, Point p) {
```

```
    // there must be at least 3 vertices in polygon[]
    if(n < 3)
      return false;

    // create a point for line segment from p to infinite
    Point extreme = new Point(INF, p.y);

    // count intersections of the above line with sides of polygon
    int count = 0, i = 0;
    do {
      int next = (i + 1) % n;

      // check if the line segment from 'p' to 'extreme' intersects with the line segment from
'polygon[i]' to 'polygon[next]'
      if(doIntersect(polygon[i], polygon[next], p, extreme)) {
        // if the point 'p' is collinear with line segment 'i-next', then check if it lies on
segment
        // if it lies, return true, otherwise false
        if(orientation(polygon[i], p, polygon[next]) == 0)
          return onSegment(polygon[i], p, polygon[next]);

        count++;
      }
      i = next;
    } while(i != 0);

    // return true if count is odd, false otherwise
    return (count % 2 == 1);
  }

  int checkAllPrecincts() {
    for(Map.Entry<Integer, Point[]> entry : precincts.entrySet()) {
      int precinct = entry.getKey();
      Point[] polygon = entry.getValue();
      if(isInside(polygon, polygon.length, point))
        return precinct;
    }
    return 0;
  }

  public static void main(String[] args) throws Exception {
    Precinct10 p = new Precinct10();
    p.input("input.txt");
    int precinct = p.checkAllPrecincts();
    p.output("output.txt", precinct);
    System.out.println("end.");
  }
}
```

Python:

```
# the explanations can be found here:
http://www.dcs.gla.ac.uk/~pat/52233/slides/Geometry1x1.pdf
# and here: https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/
# and here: https://en.wikipedia.org/wiki/Point_in_polygon

import re
```

```
# read and parse input file
def input(fname):
  precincts = {} # precinct number -> list of coordinates
  with open(fname) as in_file:
    line = in_file.readline()
    numbers_str = re.split(r"[\s,]\s*", line.strip()) # split by either white spaces or commas
    # the following will croak if the elements are not all integers
    numbers = [int(x) for x in numbers_str]
    if len(numbers) != 1:
      raise Exception("first line must have 1 number")
    n = numbers[0]

    for i in range(n):
      line = in_file.readline()
      numbers_str = re.split(r"[\s,]\s*", line.strip()) # split by either white spaces or
commas
      # the following will croak if the elements are not all floats
      numbers = [float(x) for x in numbers_str]
      precincts[int(numbers[0])] = [(numbers[i], numbers[i+1]) for i in
range(1,len(numbers),2)]

    line = in_file.readline()
    numbers_str = re.split(r"[\s,]\s*", line.strip()) # split by either white spaces or commas
    # the following will croak if the elements are not all floats
    numbers = [float(x) for x in numbers_str]
    if len(numbers) != 2:
      raise Exception("last line must have 2 numbers")
    point = (numbers[0], numbers[1])
  return n, precincts, point

def output(fname, precinct):
  with open(fname, "w") as out_file:
    out_file.writelines(f"{precinct}\n")

# given three collinear points p, q, r, the function checks if point q lies on line segment
'pr'
def on_segment(p, q, r):
  if max(p[0], r[0]) >= q[0] >= min(p[0], r[0]) and max(p[1], r[1]) >= q[1] >= min(p[1],
r[1]):
    return True
  return False

# to find orientation of ordered triplet (p, q, r)
# the function returns following values
# 0 --> p, q and r are collinear
# 1 --> clockwise
# 2 --> counter clockwise
def orientation(p, q, r):
  val = (q[1] - p[1]) * (r[0] - q[0]) - (q[0] - p[0]) * (r[1] - q[1])
  if val == 0:
    return 0  # collinear
  return 1 if val > 0 else 2 # clock or counterclock -wise

# the function that returns true if line segment 'p1q1' and 'p2q2' intersect
def do_intersect(p1, q1, p2, q2):
  # find the four orientations needed for general and special cases
  o1 = orientation(p1, q1, p2)
```

```
  o2 = orientation(p1, q1, q2)
  o3 = orientation(p2, q2, p1)
  o4 = orientation(p2, q2, q1)

  # general case
  if o1 != o2 and o3 != o4:
    return True

  # special cases
  # p1, q1 and p2 are collinear and p2 lies on segment p1q1
  if o1 == 0 and on_segment(p1, p2, q1):
    return True
  # p1, q1 and p2 are collinear and q2 lies on segment p1q1
  if o2 == 0 and on_segment(p1, q2, q1):
    return True
  # p2, q2 and p1 are collinear and p1 lies on segment p2q2
  if o3 == 0 and on_segment(p2, p1, q2):
    return True
  # p2, q2 and q1 are collinear and q1 lies on segment p2q2
  if o4 == 0 and on_segment(p2, q1, q2):
    return True
  return False # doesn't fall in any of the above cases

# returns true if the point p lies inside the polygon with n vertices
def is_inside(polygon, n, p):
  # there must be at least 3 vertices in polygon
  if n < 3:
    return False

  # create a point for line segment from p to infinite
  extreme = (1e99, p[1])

  # count intersections of the above line with sides of polygon
  count = 0
  i = 0
  while True:
    next = (i+1) % n

    # check if the line segment from 'p' to 'extreme' intersects
    # with the line segment from 'polygon[i]' to 'polygon[next]'
    if do_intersect(polygon[i], polygon[next], p, extreme):
      # if the point 'p' is collinear with line segment 'i-next',
      # then check if it lies on segment. If it lies, return true, otherwise false
      if orientation(polygon[i], p, polygon[next]) == 0:
        return on_segment(polygon[i], p, polygon[next])
      count += 1
    i = next
    if i == 0:
      break

  # return true if count is odd, false otherwise
  return (count % 2) == 1

def check_all_precincts(n, precincts, point):
  for precinct, polygon in precincts.items():
    if is_inside(polygon, len(polygon), point):
      return precinct
```

```
  return None

(n, precincts, point) = input("input.txt")
precinct = check_all_precincts(n, precincts, point) or 0
output("output.txt", precinct)
print("end.")
```