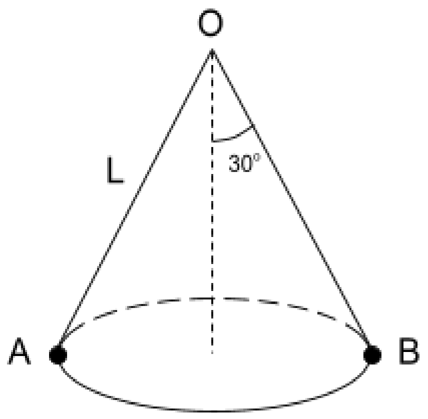
PROBLEM OF THE MONTH POM 2020 - 2021 **January, 2021**

MATHEMATICS

5 points:

What is the length of the shortest path along the side of the cone connecting points A and B that lie opposite each other on the cone's base? The length of the side of the cone OA=OB=L and the aperture of the cone is 60 degrees as shown in Figure.

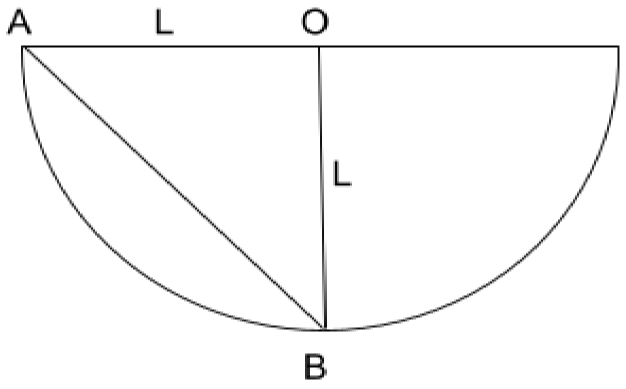


Hint:

Imagine it is a paper cone, which can be cut with the scissors.

Answer: $\sqrt{2}\,L$

Solution: The radius of the circle at the base of the cone is $L/2$ (the side of the right angle triangle opposite to the angle of 30 degrees). Its circumference is $2\pi L/2 = \pi L$. Let us imagine that the cone is made out of paper, cut it along the side OA and unfold the paper. We obtain the sector of a circle with the sector angle α such that αL is equal to the circumference of the base circle. In our case we conclude that $\alpha = \pi = 180^0$ as shown in the figure. The point B is half way along the base circle from point A and the shortest distance along the unfolded flat cone is the length AB of the straight segment connecting points A and B. We find from Pythagorean theorem $AB = \sqrt{L^2 + L^2} = \sqrt{2}\,L$.



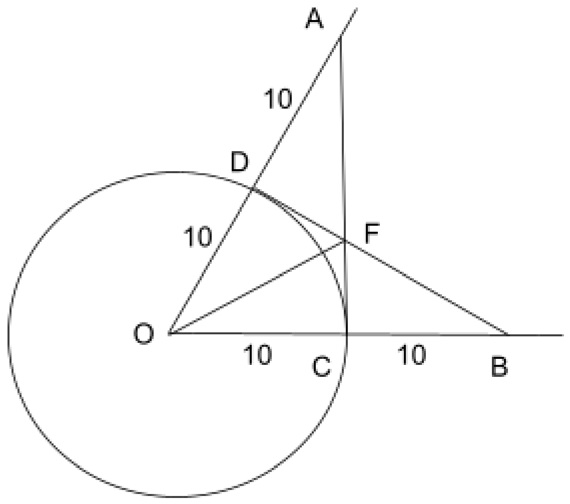
10 points:

Two straight roads OA and OB intersect at the crossroads O so that the angle AOB is 60 degrees. A runner can run along the road with velocity 20 km/h and off-road with velocity 10 km/h. What is the distance to the crossroad of the furthest point equidistant to both roads that the runner can reach in an hour?

Hint: Find all points on the plane which the runner can reach in an hour.

Answer: $20/\sqrt{3}$ km

Solution: If the runner runs off road she can reach all points inside the circle with radius OC=10 (see figure). If she goes along the road OB for the whole hour, she will reach point B such that OB=20. Assume that the runner runs part of the time along the road OB until the point B' and then runs in any direction off road. She will be able to reach all points inside some circle with the center B'. Considering all such circles for B' between O and B we conclude that the runner can reach all points inside the triangle OBD (BD is tangent to the circle with the radius OD). Similarly, if the runner runs initially along the road OA she can reach all points inside the triangle OAC. Therefore, the furthest point she can reach along the bisector of the angle AOB is the point F shown in the figure. It is easy to see that $OF = 10/cos(30) = 10/(\sqrt{3}/2) = 20/\sqrt{3}$ km.



PHYSICS

5 points: Electric car Tesla 3 has a typical range of 260 miles, and its battery stores 0.7 $kW \cdot hr$ of electric energy per liter. An alternative technology being developed for powering electric vehicles is hydrogen-based. In that case, the energy would be stored in the form of compressed hydrogen. The chemical reaction between hydrogen and atmospheric oxygen in a so-called fuel cell can directly produce electricity (without the need of burning the fuel), and the only product of that reaction is water vapor. This way, 1 mole of hydrogen gas can be converted to approximately 280 kJ of electric energy. What would be the range of the fuel cell vehicle, if the volume of its hydrogen tank is the same as that of Tesla 3 battery, and the pressure of the gas is 700 atmospheres (70 MPa)? Assume normal temperature (300 K).

Hint: Note that at normal conditions 1 mole occupies volume of approximately 22 liters.

Answer: 810 miles (the answer in the range 800-1000 miles is also acceptable)

Solution: First, consider the battery of Tesla 3. 0.7 $kW \cdot hr$ =3600 $\cdot$ 0.7 kJ= 2500 kJ is stored per 1 liter. One mole of gas occupies 22 liters at normal conditions (1 atm , 300K). Therefore, at 700 atm, there will be 700 more energy per liter. This gives $(700/22) \cdot 280\ kJ \approx 9000\ kJ$ per liter (a more accurate calculation of volume based on unified gas law gives 7800 kJ). So, the estimated range is 260 $\cdot$ (7800/2500) miles $\approx$ 810 miles (answer in the range 800-1000 miles is also acceptable).

10 points: Flywheel is a mechanical device for energy storage. It is a heavy cylinder that may be rotated at very high angular velocity, thus accumulating substantial kinetic energy. This energy is limited since the outer rim of the flywheel experiences an outward pressure due to centrifugal forces. Find the maximum kinetic energy per unit volume that the flywheel can store, if the maximum pressure that material of the rim can sustain is P. How much more (or less) energy the flywheel would store, compared to hydrogen at the same pressure P? (use results of 5 pt problem).

Hint: Consider a small element of the flywheel at distance r from the center. Its centripetal acceleration is $r\varpi^2$, where ϖ is angular velocity of the flywheel. By using the 2nd Newtons law, you may find the pressure difference across that element.

Answer: Maximum energy density is P/2, about 220 times less than hydrogen at the same pressure.

Solution: Let the flywheel be a cylinder of radius R, height H, and mass M. Consider it to be a collection of concentric rings of very small thickness Δr each. Such a ring of radius r has mass $m = 2\pi r M\Delta r/(\pi R^2) = 2rM\Delta r/R^2$ (we assume the flywheel to have a uniform density). Centripetal acceleration of all the points on the wheel is $a = r\varpi^2$, where ϖ is angular velocity of the flywheel. We now can apply 2nd Newton's law to all the points of the ring:

$$F = ma = 2r^2\varpi^2 M\Delta r/R^2$$

The force is caused by pressure difference ΔP outside and inside the ring, hence

$$\Delta P = \frac{F}{2\pi rH} = \frac{\varpi^2 M}{\pi HR^2} r\Delta r$$

The total pressure difference between the center and outer rim of the flywheel can be found as an area under the plot of linear function $\frac{\varpi^2 M}{\pi HR^2}r$ vs. r:

$$P = \frac{\varpi^2 MR^2}{2\pi HR^2} = \frac{I\varpi^2}{V} = \frac{2E}{V}$$

Here $V = \pi HR^2$ is volume, $I = \frac{MR^2}{2}$ is moment of inertia, and $E = \frac{I\varpi^2}{2}$ is kinetic energy of a rotating cylinder. Therefore, the maximum energy per volume (called energy density) is P/2. For instance, if P=70 MPa (as in 5 pt problem), one liter would store energy E=PV/2=35kJ, about 220 times less than hydrogen at the same pressure.

CHEMISTRY

This month, the topic is: **Chemical reactions, heat and energy.**
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 Sunday morning, Jan 31st. At 11:00, a Zoom conference will start where December PoM solutions will be discussed. After that, approximately at 11:30, 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

5 points:

One online textbook explains the concept of exothermic and endothermic reactions as follows:
"*When more stable substances are produced from less stable ones, such a reaction is usually exothermic; if less stable substances form from more stable ones, such a reaction is exothermic."*
What is wrong with that explanation?

Hint:

More stable state is always the state with lower energy

Answer:

To understand what is wrong with that explanation, one has to define the term "stability". The terms "energy" and "stability" are interconnected, and a state with lower energy is always more stable. Therefore, no spontaneous chemical reaction is possible where a more stable substance produces a less stable one.
However, it is necessary to keep in mind that when we speak about stability of some substance, we always specify concrete conditions. Thus, ice is more stable than liquid water at normal pressure and temperatures below zero degrees, and water is more stable at temperatures above zero degrees. Therefore, depending on the temperature, either "ice to water" or "water to ice" transitions are possible, but they are *always* a transition from a less stable to a more stable state. The "water to ice" transition is an exothermic process, but the "ice to water" transition is an endothermic process, but both those processes are transformation of a less stable substance (at those conditions) to a more stable substance.

One may argue: "If a heat is absorbed in some process, doesn't it mean it consumes energy, so the new substance has higher energy (and, therefore, less stable)?" No, that is incorrect, and the 10 point problem partially addresses that apparent paradox.

10 points:

Two compounds A and B are isomers. The compound A has a linear shape, like this:

Y-X-Y

so the angle between two Y-X bonds is 180º. In the compound B the angle between X-Y bonds is 90º.
A transforms to B in the reaction:

$$A \rightarrow B$$

Both A and B are gases at room temperature. Assuming that the energy of the X-Y bond does not change in that reaction, which process is this process more likely to be endothermic or exothermic at room temperature?
(To solve that problem, it is highly desirable to use the information that will be given in the lecture on 31st of January).

Hint:

What is the number of degrees of freedom in linear and angular molecules?

Answer:

In this problem, it is necessary to keep in mind the energy of X-Y bonds is the same in both molecules, which means both angular and linear forms have identical energy. (If some additional effects, for example, greater repulsion between "Y" atoms, were present in the angular molecule, that would affect the X-Y bonds energy, but we specified the energy stays the same). If linear and angular molecules have the same energy, can we expect to observe any thermal effect during that transition? Actually, yes. **A transition from linear to angular molecules will be endothermic**. It sounds paradoxically: the energy of each individual molecule does not change, but a heat is absorbed. How can it be possible?
Usually, that paradox is explained using the very obscure concept called *entropy*, but I will try to explain the same idea in simpler words.
To understand that paradox, let's define the term "a degree of freedom". That term refers to the number of independent types of movement which cannot be reduced to each other. Thus, an ant living in a tube can move only forward and backward, which means it has just one degree of freedom (for "backward" is "forward" times negative one, it is the same type of movement). The same ant living at the plane can move forward-back and left-right, and these are two independent types of movement. Now consider a fly. It has three degrees of freedom: it can move left-right, back-forward, and up-down, and all intermediate directions can be represented as a sum of these three elementary motions.
A single atom molecule of a gas, and similar to a fly, it has three degrees of freedom, and its velocity can be represented as a sum of three different (irreducible) principal components of motion: along x, y, and z axes, as shown at the figure below.

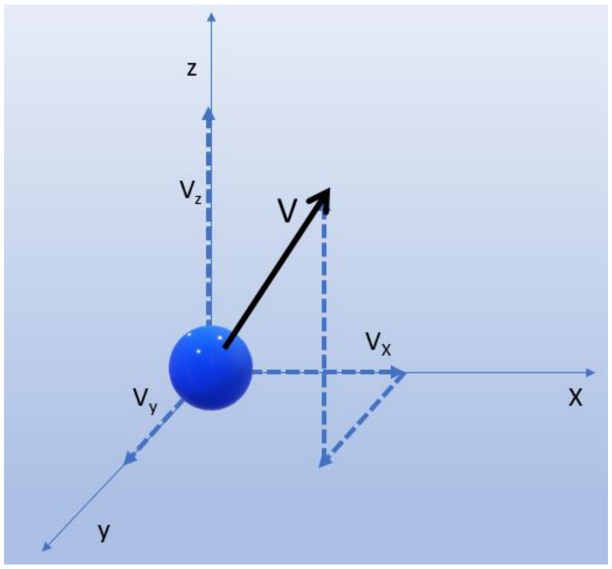


Fig. 1 Three degrees of freedom (three principal types of motion) of a single atom gas molecule.

A gas temperature is a measure of how strongly and frequently its molecules hit a thermometer's surface. But the molecule's energy that may be transferred to the thermometer depends not only on its magnitude: it also depends on its direction. Only one principal component of the velocity, concretely, the component that is perpendicular to the thermometer's surface, can be "seen" by

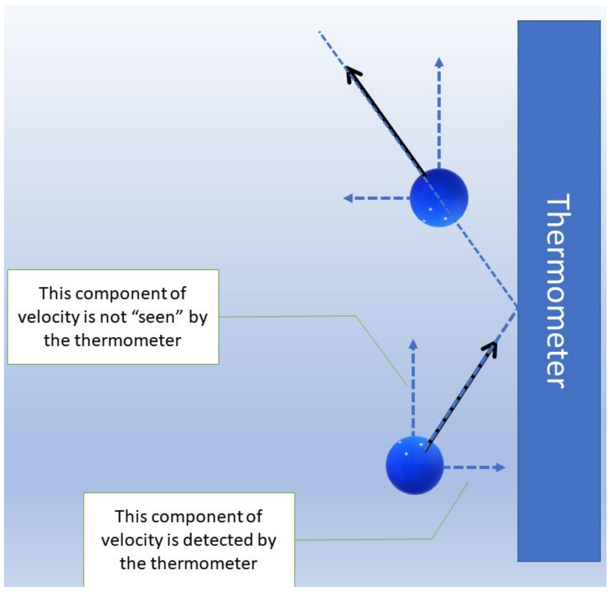

Thermometer
This component of velocity is not "seen" by the thermometer
This component of velocity is detected by the thermometer

Fig. 2. Only one principal component of velocity is "seen" by the thermometer, because others are parallel to its surface.

the thermometer (Fig 3).

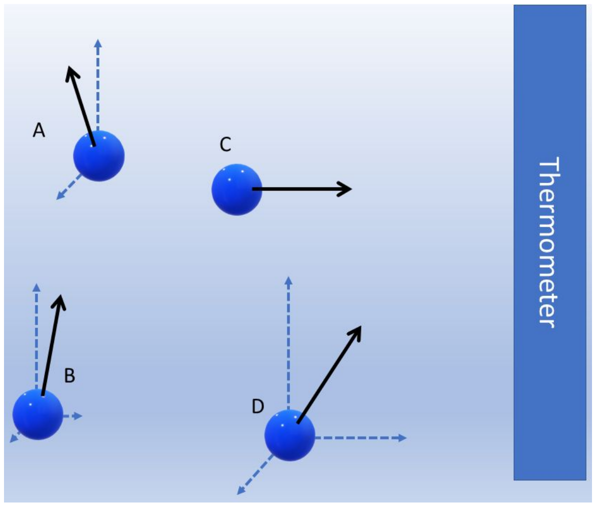


Fig. 3 The molecule C is moving perpendicularly to the thermometer's surface, so it transfers a maximal amount of energy when it hits the thermometer. The molecule A transfers no energy to the thermometer, because its velocity is parallel to the thermometer's surface. Molecules B and D transfer much less energy to the thermometer than the molecule C.

Physicists proved mathematically that when the number of molecules is big, and their motion is totally random (a gas is at equilibrium), each principal component of their velocities (v_x, v_y, and v_z), stores the same amount of kinetic energy. That means that, for example, in helium, each of its degrees of freedom contains exactly one third of the gas kinetic energy, and only one third of that kinetic energy is available for temperature measurement.
Now consider a situation when gas molecules have a more complex shape. An example is hydrogen (H_2). A hydrogen molecule is a linear molecule, so two additional degrees of freedom are available for it (rotations around two axes perpendicular to the hydrogen's main axis, see the figure 4).

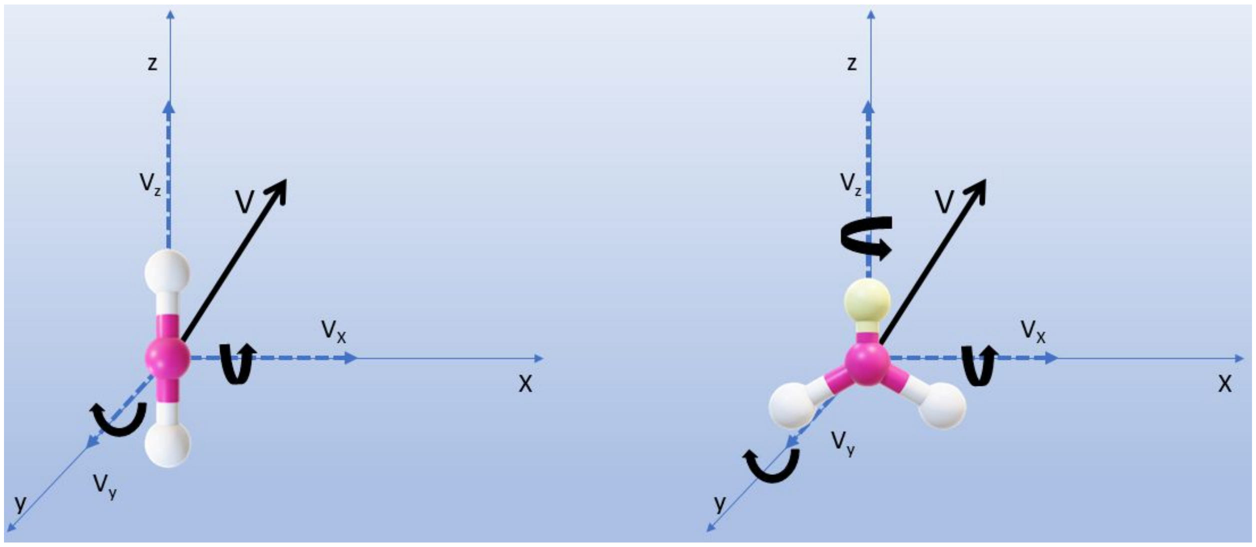


Figure 4. Degrees of freedom of a linear (left) and non-linear (right) molecules.

For trigonal and more complex molecules, one more rotational degree of freedom become possible: the molecule may rotate along x, y, and z axis. Therefore, **linear molecules have 5 degrees of freedom, and more complex molecules have 6 degrees of freedom.**

It was proven mathematically that all degree of freedom (not only translational ones) store the same amount of energy. That means in a diatomic gases (hydrogen, oxygen etc) each degree of freedom contains one fifths of the total energy, whereas the gases with non-linear molecules (ammonia, methane, hydrogen sulfide) one degree of freedom contains *one sixths* of the total kinetic energy. Remember that the thermometer “sees” the energy that is stored in only one degree of freedom (the degree of freedom that is perpendicular to the thermometer’s surface). That means that for a single atom gas (e.g. helium) one third of total kinetic energy is detected as a temperature, for diatomic gases (hydrogen, oxygen) only one fifth of gas kinetic energy can be measured as temperature, and for the gases with non-linear molecules the temperature is a measure of one sixth of its total kinetic energy.
Now let’s take a look at what happens during our reaction. The linear molecule X-Y-X transforms to an angular molecule. If the total kinetic energy of the starting material it E_{kin}, the temperature was proportional to $E_{kin}/5$ (because each molecule has five degrees of freedom: motion along X, Y, and Z axes and rotation in two directions). During the transformation to the angular molecule, no energy is produced, so the total kinetic energy (E_{kin}) stays the same. However, after the transformation, the temperature becomes equal to $E_{kin}/6$ (now each molecule rotates in *three* directions). Since the numerator in that fraction stays the same (the energy does not change), we will see a drop of temperature. Therefore, this reaction will be endothermic.

This explanation can be expanded to much more complex reactions and processes.

BIOLOGY

This month, the topic is: **Biology and topology**
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 Biology PoMs, although it is not supposed to provide direct hints.
This month, the lecture will be on Saturday, January 30. At 13:00, a Zoom conference where we will tell you about DNA topology and the enzymes that cause topological transformations in DNA (so called topoisomerases)..
To join the Zoom conference, use this link:
https://us02web.zoom.us/j/4817690592?pwd=T2djSjRETEpDSHFZdWJpYlBTYzdjQT09
Meeting ID: 481 769 0592
Passcode: 879615

5 points:

Bob was involved in a study that required preparation of a significant amount of a certain circular DNA (a.k.a. "plasmid"). This type of DNA is usually grown in bacteria, and they can be prepared in a pure form. When prepared in optimal conditions, they look like relaxed loops (as shown at the Fig. 1).

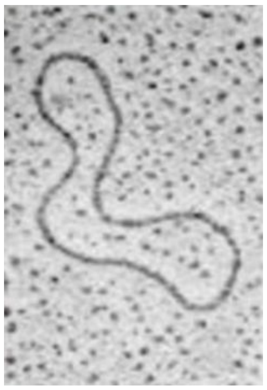


Fig. 1. An electron micrograph of circular DNA

However, when Bob checked a sample of the plasmid he prepared, the electron micrographs of his DNAs looked as shown at Fig 2 and 3: they formed knots and/or concatenated rings.

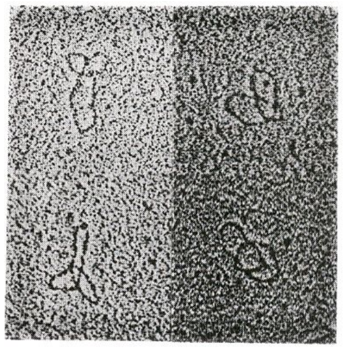


Fig 2. DNA knots

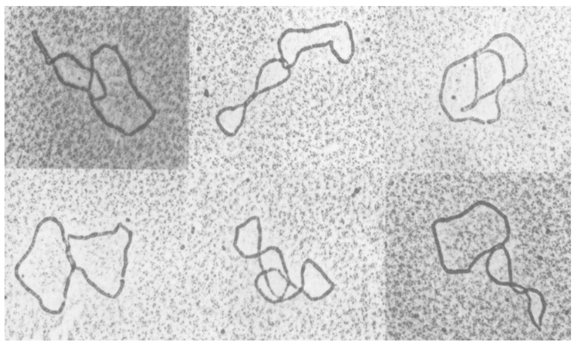


Fig. 3. Concatenated DNA rings

Bob was very disappointed, and he asked his friend Alice if she knew what was the reason. "It looks like one important enzyme is not working properly in your bacteria. That enzyme is called "topoisomerase"", she answered.

"I don't think so," - Bob argued. "If topoisomerase were not working in my bacteria, there would be no DNA replication at all, and I wouldn't be able to obtain any plasmid at all."
"Bob, you are working on a PhD in biology, and you always should keep in mind that living cells are much more complex than people usually think. In reality, there are …"
Continue Alice's words. What did she mean?

Hint:

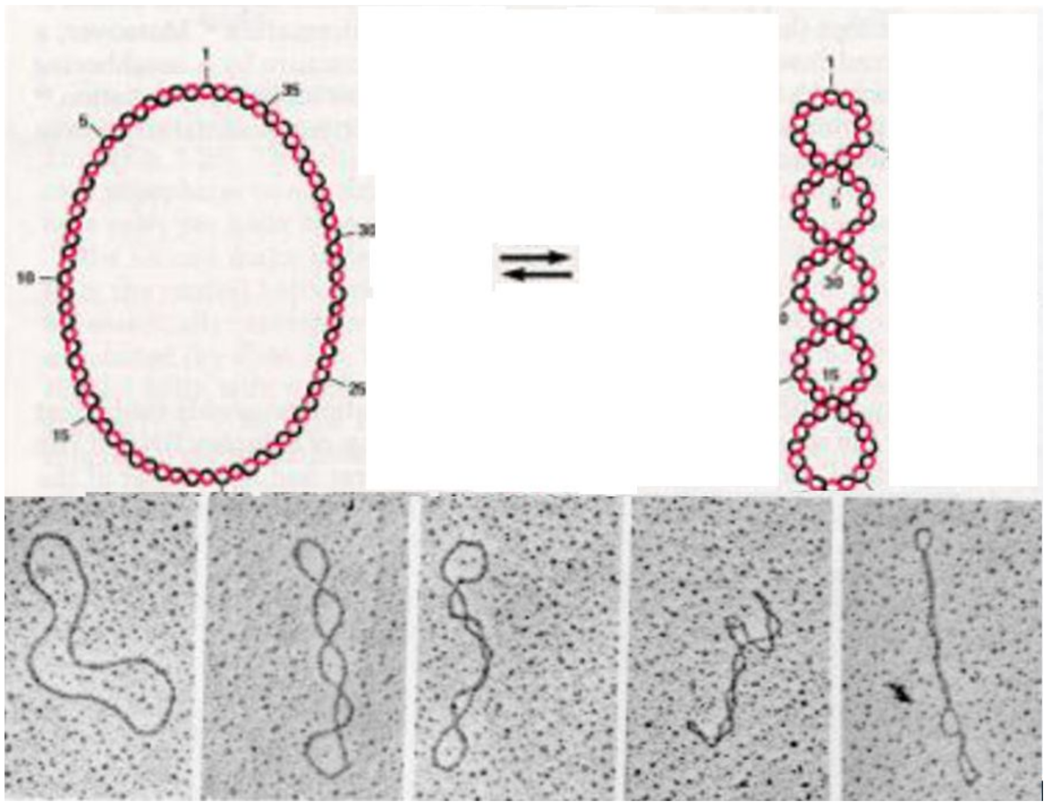
"… two main groups of DNA topoisomerase" (further continuation will be the answer).

Answer:

"... the first type topoisomerases ("topoisomerase I") break just one strand of DNA's double helix. By breaking one strand, the topoisomerase creates a temporary "hinge", or "swivel", so the DNA can freely rotate, so undertwisted or overtwisted DNA helixes are easily relaxed. However, if a DNA forms a knot, or two circular DNA are connected like rings is a chain, breakage of just one DNA strand cannot resolve that problem. That task is performed by the second group of DNA topoisomerases (so called "topoisomerase II"): they break *both* DNA strands and allow a second segment of DNA to pass through the gap. After that, the gap is sealed back. It seems everything is ok with topoisomerase I is your bacteria, but topoisomerases II are not working properly"
That is what Alice said.

10 points:

Usually, it is considered that one of the main functions of DNA topoisomerases is to untwist the DNA that becomes twisted during the DNA replication process. In other words, they convert twisted DNA as shown at the figure below (the series of electron micrographs of differently twisted DNA are shown at the bottom, and most topoisomerases from right to left):



,

Relaxes ⟷ Moderately twisted ⟷ Highly twisted

However, a group of DNA topoisomerases exist that work in an opposite direction: they conwert a relaxed (untwisted) DNA into a twisted form (the forms similar to the one shown at the right side of the picture). Is there any physiological reason for that? Explain.

.

Hint:

Some important processes, such as transcription, require local melting of DNA (in that context, “melting” means local separation of DNA strands, which happens via opening base pairs without breakage of the DNA’s backbone).

Answer:

In living cells, many processes, such as DNA replication, DNA transcription, or chromatin packaging lead to a moderate overtwisting of the DNA helix. “Overtwisting” means that instead of making one turm per 10 nucleotide base pairs, the helix makes one turm each 9.5 to 9.8 nucleotides. That may happen, for example, when the DNA is being packed into a dense superhelix in chromatin, or when DNA polymerase is moving along DNA (you may try to experiment with a twisted rope, and you will easily see it). An overtwisted DNA ring in a free state forms a dense superhelix (shown at the above figure at the right panel) to compensate for the helix’s strain. DNA can also be “undertwisted”, which means the DNA helix makes a full turn each 10.2 to 10.5 nucleotides. A free undertwisted DNA also looks as a superxelix, but the direction of the supercoil is opposite.

The enzymes called “DNA topoisomerases” usually help the overtwisted DNA to relax, but it is more convenient for a cell to keep DNA slightly undertwisted all the time. Such an undertwisted DNA is easier to replicate, transcribe, and it forms superhelical structures in chromatine more easily. In bacteria, that undertwisting is usually performed by the enzymes called “DNA gyrases”, which belong to the Topoisomerase II class enzymes. They are working against equilibrium, so they consume ATP to make DNA undertwisted. In eukaryotes, a situation is more complex, and different mechanisms exist..

LINGUISTICS

5 points:

Consider the following sentences and their translations from an unnamed language.

Karapo muba	"The dog eats"
Nerip turio	"The cat sleeps"
Karapo tarak hisa	"The man eats a lot"
Karapo hisa babon	"The old man eats"
Karapo turio tarak	"The tiger eats"
Soret babon muba kiun	"The puppy walks for a long time"
Karapo hams muba	“The bear eats the dog”

Question 1: Translate the following sentences into this language. Make educated guesses about the words that you have not encountered in the given data.

1) The kitten eats a little
2) The wolf eats a lot
3) The tiger sleeps for a long time
4) The chief eats a lot of tiger

Question 2: Translate the following sentences into English

1) *Nerip babon hisa kiun*
2) *Soret tarak turio babon*

Answer:

Question 1:

"The kitten eats a little" = eats small cat small = Karapo kiun turio kiun.

"The wolf eats a lot" = eats big dog big = Karapo tarak muba tarak.
"The tiger sleeps for a long time" = sleeps old cat big = Nerip babon turio tarak.
“The chief eats a lot of tiger” = eats big man big cat big = Karapo tarak hisa tarak turio tarak

Question 2:

sleeps old man small = nerip babon hisa kiun "The boy sleeps for a long time"
walks big cat old = soret tarak turio babon "The old cat walks a lot"

Solution:

Comparing the data sentences we can determine the words, and the order of words: Verb-Subject-Object. In addition, the adjective follows the noun and adverbs follows the verb. Another important part of the grammar to realize here is that adjectives and adverbs are denoted by the same words, that it “small” = “a little”; “big” = “a lot”; “old = “for a long time”. In addition, by comparing the number of words, we can see that “tiger” is “big cat”, kitten is “small cat”, and puppy is “small dog”. This logic should allow us to translate the words we did not encounter before: “wolf” is “big dog”, “chief” is “big man”.

eats = karapo
sleeps = nerip
walks = soret

dog = muba
cat = turio
man = hisa

big/a lot = tarak
old/for a long time = babon
small/a little = kiun

eats dog = karapo muba "The dog eats"
sleeps cat = nerip turio "The cat sleeps"
eats big man = karapo tarak hisa "The man eats a lot"
eats man old = karapo hisa babon "The old man eats"
eats cat big = karapo turio tarak "The tiger eats"
walks old dog small = soret babon muba kiun "The puppy walks for a long time"

10 points:

Below is a set of artificial sentences:

She took the glove off from her 1.

Her 1 was covered with a hat.
Around her 1, she tied a scarf.

In this case, the number 1 is a placeholder for a word meaning "the body part on which the mentioned article of clothing is usually worn". The words "hand", "head", and "neck" are assumed to be replaced by the number 1 in the sentences above, respectively. Knowing the definition of the numerical placeholder, it's easy to determine the meaning of the sentence.

Try to come up with as many categories of replaceable words as you can in the sentences below, and substitute them with numbers. Then, **describe what type of words your placeholder numbers represent**. Keep in mind that anyone should be able to understand the sentences below if they know the meaning of the numbers you choose.

He made a horrible mess.
The cow mooed, while her calf whined and butted against me.
She eradicated the deadly disease.
His song waned.
The cat meowed.
My aunt knitted the boy a cozy sweater.
A blazing light shone.
Warm showers rained on us.
They cancelled their immediate order.
You started the great schism!
The light dissipated.
The powerful rain stopped, and the heat wave came.
She built a house.
The strong wind blew.
The horse neighed and kicked, just like its chestnut foal.
She demolished her house.
A flame ignited.
The professor yelled as he argued.
He designed an infographic.

Answer:

This problem allows for many answers, here is one possibility:

He 3 a 7 mess.
The cow 5, while 8 9 5 and 6 against me.
She 4 the 7 disease.
8 song 2.
The cat 5.

8 aunt 3 the 9 a cozy sweater.
A 7 light 0.
Warm showers 0 on us.
They 4 8 7 order.
You 3 the 7 schism!
The light 2.
The 7 rain 2, and the heat wave 1.
She 3 a house.
The 7 wind 0.
The horse 5 and 6, just like 8 chestnut 9 .
She 4 8 house.
A flame 1.
The professor 5 as he 6.
He 3 an infographic.

0 - verb implying existence; 1 - verb implying the coming into existence; 2 - verb implying the end of existence; 3 - verb implying the creation of something; 4 - verb implying the destruction of something; 5 - verb implying the production of a sound; 6 - verb implying an aggressive action; 7 - strong descriptive adjective; 8 - possessive article; 9 - young animal

Solution:

See Answer above.

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 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:

Sigma Camp decided to come up with its own flag. They decided to make the flag a constellation of σ signs, as many as there are campers admitted to the camp, arranged in an aesthetically appealing rectangular formation, just like stars in the American flag. Formation could have the same number of signs in each row, just like the USA flag circa 1912, for example:

```
σ σ σ σ σ σ
σ σ σ σ σ σ
σ σ σ σ σ σ
σ σ σ σ σ σ
```

or alternating number of stars in each row: n, n-1, n, n-1.... As in the current USA flag. For example:

```
σ σ σ σ σ σ σ
 σ σ σ σ σ σ
σ σ σ σ σ σ σ
 σ σ σ σ σ σ
σ σ σ σ σ σ σ
```

To be aesthetically appealing , the aspect ratio of the star pattern (ratio of the number of columns to the number of rows) must be between 1.3 and 1.8.

Your program will receive the number of admitted campers as an input in **input.txt** file. It should write the output to output.txt file in the following format:
Each row in the flag will be represented by a number of σ signs in the corresponding row of the flag. For example, the flags above would be represented as:

6 6 6 6
and
7 6 7 6 7
correspondingly.

If the number of admitted campers cannot be represented in an aesthetically appealing flag, the program should write IMPOSSIBLE to output.txt.

Solution:

Java:

```
/*
Note: we assume that only 2 cases shown in the problem are admissible, and we will not
consider when the last row ends in "n-1" stars.
Let k be the number of campers (total number of stars), m be the number of rows and n be the
number of columns.
k, m, n are natural numbers.
Let's consider each case separately.
1) m*n = k                                                                   (1)
   1.3 <= n/m <= 1.8                                                         (2)
   From (1) => n = k/m                                                       (3)
   Substituting this into (2) yields
   1.3*m <= k/m <= 1.8*m
   sqrt(k/1.8) <= m <= sqrt(k/1.3)                                           (4)
   So, we'll try all integers satisfying (4) and for which (3) also gives an integer n.
   If we find nothing then we'll consider the 2nd case.
2) [n + (n-1)] * (m-1)/2 + n = k
   (2n-1)*(m-1) + 2n = 2k
   2nm - m - 2n + 1 + 2n = 2k
   m*(2n - 1) + 1 = 2k
   m = (2k - 1) / (2n - 1)                                                   (5)
   Substitute this into (2) to obtain
   1.3*(2k - 1) / (2n - 1) <= n <= 1.8*(2k - 1) / (2n - 1)
   Let's consider first the left inequality.
   2.6k - 1.3 <= 2n^2 - n
   2n^2 - n - 2.6k + 1.3 >= 0
   Since n is a natural number we get
   n >= [1 + sqrt(1 - 4*2*(-2.6k+1.3))] / 4
     >= [1+sqrt(20.8k-9.4)] / 4
   Now consider the right inequality:
   n * (2n - 1) <= 1.8*(2k - 1)
   2n^2 - n - 3.6k + 1.8 <= 0
   1 <= n <= [1+sqrt(1-4*2*(-3.6k+1.8))] / 4
          <= [1+sqrt(28.8k-13.4)] / 4
   Thus
   [1 + sqrt(20.8k-9.4)] / 4 <= n <= [1 + sqrt(28.8k-13.4)] / 4      (6)
   i.e. we'll search for all integers in (6) for which (5) also produce an integer m.
   In case of ambiguity we prefer the layout #1.
*/

import java.io.*;
import java.util.List;
```

```
public class SigmaFlag5 {
  int k, m, n;

  void input(String fname) throws Exception {
    try(BufferedReader br = new BufferedReader(new FileReader(fname))) {
      String line = br.readLine().trim();
      k = Integer.parseInt(line); // will throw exception if not integer
    }
  }

  boolean case1() {
    for(m=(int)Math.ceil(Math.sqrt(k/1.8)); m<=(int)Math.floor(Math.sqrt(k/1.3)); m++) {
      double n2 = (double)k / m;
      if(Math.abs(n2 - (int)n2) < 1e-8) {
        n = (int)n2;
        return false;
      }
    }
    return true;
  }

  boolean case2() {
    for(n=(int)Math.ceil((1 + Math.sqrt(20.8*k-9.4)) / 4);
        n<=(int)Math.floor((1 + Math.sqrt(28.8*k-13.4)) / 4); n++) {
      double m2 = (2*(double)k - 1) / (2*n - 1);
      if(Math.abs(m2 - (int)m2) < 1e-8) {
        m = (int)m2;
        return false;
      }
    }
    return true;
  }

  void output1(String fname, String msg) throws IOException {
    try(BufferedWriter bw = new BufferedWriter(new FileWriter(fname))) {
      bw.write(String.format("%s\n", msg));
    }
  }

  void output2(String fname, int pattern) throws IOException {
    try(BufferedWriter bw = new BufferedWriter(new FileWriter(fname))) {
      if(pattern == 1)
        for(int i=0; i<m; i++)
          bw.write(String.format("%d ", n));
      else { // pattern 2
        for(int i=0; i<(int)Math.floor((double)m/2); i++)
          bw.write(String.format("%d %d ", n, n-1));
        bw.write(String.format("%d", n));
      }
      bw.write("\n");
    }
  }

  public static void main(String[] args) throws Exception {
    SigmaFlag5 flag = new SigmaFlag5();
    flag.input("input.txt");
    boolean err = flag.case1();
    int pattern = 1;
    if(err) {
      err = flag.case2();
      pattern = 2;
    }
    if(err)
      flag.output1("output.txt", "IMPOSSIBLE");
    else
      flag.output2("output.txt", pattern);
    System.out.println("end.");
  }
}
```

Python-3:

```
"""
Note: we assume that only 2 cases shown in the problem are admissible, and we will not
consider when the last row ends in "n-1" stars.
Let k be the number of campers (total number of stars), m be the number of rows and n be the
number of columns.
k, m, n are natural numbers.
Let's consider each case separately.
1) m*n = k                                                                (1)
   1.3 <= n/m <= 1.8                                                      (2)
   From (1) => n = k/m                                                    (3)
   Substituting this into (2) yields
   1.3*m <= k/m <= 1.8*m
   sqrt(k/1.8) <= m <= sqrt(k/1.3)                                        (4)
   So, we'll try all integers satisfying (4) and for which (3) also gives an integer n.
   If we find nothing then we'll consider the 2nd case.
2) [n + (n-1)] * (m-1)/2 + n = k
   (2n-1)*(m-1) + 2n = 2k
   2nm - m - 2n + 1 + 2n = 2k
   m*(2n - 1) + 1 = 2k
   m = (2k - 1) / (2n - 1)                                                (5)
   Substitute this into (2) to obtain
   1.3*(2k - 1) / (2n - 1) <= n <= 1.8*(2k - 1) / (2n - 1)
   Let's consider first the left inequality.
   2.6k - 1.3 <= 2n^2 - n
   2n^2 - n - 2.6k + 1.3 >= 0
   Since n is a natural number we get
   n >= [1 + sqrt(1 - 4*2*(-2.6k+1.3))] / 4
     >= [1+sqrt(20.8k-9.4)] / 4
   Now consider the right inequality:
   n * (2n - 1) <= 1.8*(2k - 1)
   2n^2 - n - 3.6k + 1.8 <= 0
   1 <= n <= [1+sqrt(1-4*2*(-3.6k+1.8))] / 4
          <= [1+sqrt(28.8k-13.4)] / 4
   Thus
   [1 + sqrt(20.8k-9.4)] / 4 <= n <= [1 + sqrt(28.8k-13.4)] / 4      (6)
   i.e. we'll search for all integers in (6) for which (5) also produce an integer m.
   In case of ambiguity we prefer the layout #1.
"""

import math

# read and parse input file
```

```
def input(fname):
  with open(fname) as in_file:
    line = in_file.readline().strip()
    return int(line) # will throw exception if not integer

def case1(k):
  for m in range(math.ceil(math.sqrt(k/1.8)), math.floor(math.sqrt(k/1.3))+1): # +1 because of
                                                                                # range()
    n = k / m # python gives float division
    if n.is_integer():
      return False, m, int(n)
  return True, None, None

def case2(k):
  for n in range(math.ceil((1 + math.sqrt(20.8*k-9.4)) / 4), math.floor((1 +
                 math.sqrt(28.8*k-13.4)) / 4) +1): # +1 because of range()
    m = (2*k - 1) / (2*n - 1) # python gives float division
    if m.is_integer():
      return False, int(m), n
  return True, None, None

def output1(fname, msg):
  with open(fname, "w") as out_file:
    out_file.writelines(f"{msg}\n")

def output2(fname, m, n, pattern):
  with open(fname, "w") as out_file:
    if pattern == 1:
      for i in range(m):
        out_file.writelines(f"{n} ")
    else: # pattern 2
      for i in range(math.floor(m/2)):
        out_file.writelines(f"{n} {n-1} ")
      out_file.writelines(f"{n}")
    out_file.writelines("\n")

k = input("input.txt")
err, m, n = case1(k)
pattern = 1
if err:
  err, m, n = case2(k)
  pattern = 2
if err:
  output1("output.txt", "IMPOSSIBLE")
else:
  output2("output.txt", m, n, pattern)
print("end.")
```

10 points:

A group of Sigma counselors was working on turning a map of the Silver Lake campus into a simple jigsaw puzzle. To do that, they split the rectangular map of the campus into NxM square pieces and cut the map into those little squares. The counselors had to break their activity to attend a staff meeting. At that time some pranksters decided to meddle with jigsaw making and mixed all the squares into a pile. They did not know that counselors, being smart, anticipated such meddling and meticulously marked all the squares the following way:

- they marked the connections for each square, recording connections on top, right, bottom, left
- connections were marked as pairs of corresponding positive/negative numbers, 1/-1 to 10/-10, so that if one side of the connection would be marked *k* the opposite side of the connection would be marked *-k.* Therefore 1 would connect to -1, 2 to -2, etc.
- assignment of the connection number pairs was somewhat arbitrary, but guaranteed that each square had a unique “connection signature”
- lack of connection to another square (for edge squares) was denoted by number 0

Your program will receive its input from the file **input.txt** of the following structure:

The first row will contain 2 space-separated numbers N and M denoting the number of rows and columns the map has been split into.
This is followed by N*M rows, each containing a definition of a square in the following format: a square number, from 0 to (N*M-1), immediately followed by a colon (':') and then followed by 4 comma-separated integers denoting up, right, down, left connections.

Assume squares are rotated in the correct direction.
An example of input file is :

```
2 2
0:0,1,2,0
1:0,0,2,-1
2:-2,0,0,2
3:-2,-2,0,0
```

Your program needs to help the counselors to reassemble the map. Its output should be written to **output.txt** file, which would contain N rows of M space-separated numbers of squares in the right sequence.

For the example above, the output file would contain:

```
0 1
3 2
```

Solution:

Java:

```
/*
We'll build up the puzzle row by row starting from the top left corner. If there are multiple
candidates for a piece, we'll try them all (backtracking).
*/
```

```
import java.io.*;
import java.util.*;

public class SigmaPuzzle10 {
  public final static int TOP    = 0;
  public final static int RIGHT  = 1;
  public final static int BOTTOM = 2;
  public final static int LEFT   = 3;
  int m = 0; // number of rows
  int n = 0; // number of columns
  Map<Integer, int[]> pieces = new HashMap<>(); // pieces of puzzle

  void input(String fname) throws Exception {
    try(BufferedReader br = new BufferedReader(new FileReader(fname))) {
      String line = br.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 != 2)
        throw new Exception("first line must have 2 numbers");
      m = numbers[0];
      n = numbers[1];
      if(m<1 || n<1)
        throw new Exception("m and n must be natural numbers");
      for(int i=0; i<m*n; i++) {
        line = br.readLine().trim();
        nums = line.split("\\s*:\\s*");
        if(nums.length != 2)
          throw new Exception("a line must have exactly 1 ':'");
        int id = Integer.parseInt(nums[0]);
        nums = nums[1].split("\\s*[\\s,]\\s*");
        if(nums.length != 4)
          throw new Exception("must have 4 connections");
        numbers = Arrays.stream(nums).map(s ->
                          Integer.valueOf(s)).mapToInt(Integer::intValue).toArray();
        pieces.put(id, numbers);
      }
    }
  }

  // find pieces of puzzle matching the left and top sides from a bag identified by "ids"
  List<Integer> findPieces(int left, int top, Set<Integer> ids) {
    List<Integer> res = new ArrayList<>();
    for(int id : ids) {
      int[] piece = pieces.get(id);
      if(piece[LEFT]==-left && piece[TOP]==-top)
        res.add(id);
    }
    return res;
  }

  // given left and top pieces
  boolean putNextPiece(int n, int[] prevBottom, int prevRight, Set<Integer> available,
List<Integer> curRow, List<List<Integer>> solution) {
    if(available.size() == 0) {
      System.out.print("row complete: ");
      curRow.forEach(x -> System.out.printf("%d ", x));
      System.out.println();
      solution.add(curRow);
      System.out.println("puzzle complete");
      return true;
    }
    if(n == this.n) {
      System.out.print("row complete: ");
      curRow.forEach(x -> System.out.printf("%d ", x));
      System.out.println();
      // for(int i=0; i<n; i++)
      //   prevBottom[i] = pieces.get(curRow.get(i))[BOTTOM];
      prevBottom = curRow.stream().map(x ->

pieces.get(x)[BOTTOM]).mapToInt(Integer::intValue).toArray();
      solution.add(curRow);
      if(putNextPiece(0, prevBottom, 0, available, new ArrayList<>(), solution))
        return true;
      solution.remove(curRow);
      return false;
    }
    List<Integer> candidates = findPieces(prevRight, prevBottom[n], available);
    for(int candidate : candidates) {
      Set<Integer> available2 = new HashSet<>(available);
      available2.remove(candidate);
      curRow.add(candidate);
      System.out.printf("trying <%d> at col %d\n", candidate, n);
      if(putNextPiece(n+1, prevBottom, pieces.get(candidate)[RIGHT], available2, curRow,
                      solution))
        return true;
      curRow.remove(candidate);
    }
    return false;
  }

  boolean solve(List<List<Integer>> solution) {
    int[] prevBottom = new int[n];
    for(int i=0; i<n; i++)
      prevBottom[i] = 0;
    // build the puzzle like a snake: column by column and row by row
    int prevRight = 0;
    // top-leftmost
    List<Integer> candidates = findPieces(prevRight, prevBottom[0], pieces.keySet());
    if(candidates.size() != 1)
      return false;
    System.out.printf("adding <%d> at col 0\n", candidates.get(0));
    Set<Integer> available = new HashSet<>(pieces.keySet());
    available.remove(candidates.get(0));
    List<Integer> curRow = new ArrayList<>();
    curRow.add(candidates.get(0));
    boolean res = putNextPiece(1, prevBottom, pieces.get(candidates.get(0))[RIGHT], available,
                               curRow, solution);
    return res;
  }

  void output(String fname, boolean res, List<List<Integer>> solution) throws IOException {
    try(BufferedWriter bw = new BufferedWriter(new FileWriter(fname))) {
      if(res) {
        for(List<Integer> row : solution) {
          for(int col : row) {
            bw.write(String.format("%d ", col));
          }
          bw.write("\n");
        }
      }
      else
        bw.write("impossible\n");
    }
  }

  public static void main(String[] args) throws Exception {
    SigmaPuzzle10 puzzle = new SigmaPuzzle10();
    puzzle.input("input.txt");
    List<List<Integer>> solution = new ArrayList<>();
    boolean res = puzzle.solve(solution);
    if(res) {
      System.out.println("puzzle solved:");
      solution.forEach(row -> { row.forEach(col -> System.out.printf("%d ", col));
      System.out.println();});
    }
    else
      System.out.println("puzzle cannot be solved");
    puzzle.output("output.txt", res, solution);
    System.out.println("end.");
  }
}
```

Python-3:

```
"""
We'll build up the puzzle row by row starting from the top left corner. If there are multiple
candidates for a piece, we'll try them all (backtracking).
"""

import re

class Puzzle:
  TOP    = 0
  RIGHT  = 1
  BOTTOM = 2
  LEFT   = 3
  m = 0 # number of rows
  n = 0 # number of columns
  pieces = {} # pieces of puzzle

  # read and parse input file
  def input(self, fname):
    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) != 2:
        raise Exception("first line must have 2 numbers")
      self.m = numbers[0]
      self.n = numbers[1]
      if self.m < 1 or self.n < 1:
        raise Exception("m and n must be natural numbers")

      for i in range(self.m * self.n):
        line = in_file.readline().strip()
        numbers_str = re.split(r"\s*:\s*", line)
        if len(numbers_str) != 2:
          raise Exception("a line must have exactly 1 ':'")
        id = int(numbers_str[0])
        numbers_str = re.split(r"[\s,]\s*", numbers_str[1])
        if len(numbers_str) != 4:
          raise Exception("must have 4 connections")
        numbers = [int(x) for x in numbers_str]
        self.pieces[id] = numbers

  # find pieces of puzzle matching the left and top sides from a bag identified by "ids"
  def find_pieces(self, left, top, ids):
    res = []
    for id in ids:
      piece = self.pieces[id]
      if piece[self.LEFT]==-left and piece[self.TOP]==-top:
        res.append(id)
    return res

  # given left and top pieces
  def put_next_piece(self, n, prev_bottom, prev_right, available, cur_row, solution):
    if len(available) == 0:
      print(f"row complete: {cur_row}")
      solution.append(cur_row)
      print(f"puzzle complete")
      return True
    if n == self.n:
      print(f"row complete: {cur_row}")
      prev_bottom = [self.pieces[x][self.BOTTOM] for x in cur_row]
      solution.append(cur_row)
      if self.put_next_piece(n=0, prev_bottom=prev_bottom, prev_right=0, available=available,
                             cur_row=[], solution=solution):
        return True
      solution.remove(cur_row)
      return False
    candidates = self.find_pieces(prev_right, prev_bottom[n], available)
    for candidate in candidates:
      available2 = available.copy()
      available2.remove(candidate)
      cur_row.append(candidate)
      print(f"trying <{candidate}> at col {n}")
      if self.put_next_piece(n+1, prev_bottom, self.pieces[candidate][self.RIGHT], available2,
                             cur_row, solution):
        return True
      cur_row.remove(candidate)
    return False

  def solve(self):
    prev_bottom = [0 for _ in range(self.n)]
    # build the puzzle like a snake: column by column and row by row
    prev_right = 0
    candidates = self.find_pieces(prev_right, prev_bottom[0], self.pieces.keys()) #
top-leftmost
    if len(candidates) != 1:
      return False
    print(f"adding <{candidates[0]}> at col 0")
    available = list(self.pieces.keys())
    available.remove(candidates[0])
    cur_row = [candidates[0]]
    solution = []
    res = self.put_next_piece(1, prev_bottom, self.pieces[candidates[0]][self.RIGHT],
available,
                              cur_row, solution)
    return res, solution

  def output(self, fname, res, solution):
    with open(fname, "w") as out_file:
      if res:
        for row in solution:
          for col in row:
            out_file.write(f"{col} ")
          out_file.write("\n")
      else:
        out_file.writelines("impossible\n")

if __name__ == '__main__':
  puzzle = Puzzle()
  puzzle.input("input.txt")
  res, solution = puzzle.solve()
  if res:
    print("puzzle solved:\n", solution)
  else:
    print("puzzle cannot be solved")
  puzzle.output("output.txt", res, solution)
  print("end.")
```