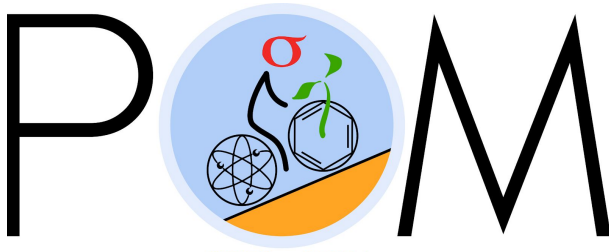
PROBLEM OF THE MONTH



September, 2020

2020 - 2021

MATHEMATICS

5 points:

Consider an infinite 3-dimensional cubic lattice with black nodes at the vertices of the cubes and in the center of every face and white nodes on the midpoint of every edge. What is the ratio of the number of black nodes to white nodes?

Hint: Consider each node in a single cubic cell and think how many cubes share that node.

Answer: 4/3

Solution: Let us consider a single cube and count the number of black nodes to be assigned to this cube. We have 8 black nodes in vertices of the cube. Each vertex is shared between 8 cubes so the number per given cube is 8/8=1. In addition, we have 6 nodes in the center of each face with each face shared between two cubes so the number is 6/2=3. Therefore, the number of black nodes associated with a single cubic cell of the lattice is 8/8+6/2=4. Analogously, the number of white nodes is 12/4=3 (12 edges, each shared between 4 cubes). Now we easily compute the ratio of black nodes to white nodes 4/3.

10 points:

A cube with an integer side *n* is divided using barriers into 1x1x1 cubes. How many barriers do you need to break down in order to be able to get from any 1x1x1 cube to the edge?
(A single barrier is one face of a 1x1x1 cube)

Hint: Find out how the number of disconnected domains changes when one breaks an internal barrier.

Answer: $(n-2)^3$

Solution: Let us remove all outside faces of the large cube. As a result we obtain $(n-2)^3+1$ disconnected parts of space (the outside of the cube plus all insides of remaining 1x1x1 cubes). Assume that the necessary number of barriers has been broken and one can get to the edge of the large cube from inside any of 1x1x1 cubes. As a result with outside faces still removed we obtain a single fully connected domain (one can get from any point of the space to any point without crossing a barrier. Breaking one barrier always decreases the number of connected components by 1 or 0. It means that to go from $(n-2)^3+1$ components to 1 component one needs to break at least $(n-2)^3$ barriers.

It is easy to show that this minimum can be achieved. Indeed, let us break all bottom sides of $(n-2)^3$ internal cubes. This way one can reach the bottom face of a big cube from any small cube.

PHYSICS

5 points: An SUV weighing **2000 kg** and moving with a speed of **15 m/s** fails to stop at a stop sign on an intersection. It hits a sedan weighing **1000 kg** and moving in a perpendicular direction at a speed of **21 m/s**. The collision is completely ***inelastic***: cars stick and continue moving together. How far from the collision point will they come to rest? The coefficient of kinetic friction between the rubber tires and the asphalt is $\mu = 0.7$ (assume that both drivers were applying brakes at the moment of the collision).

Hint: Only momentum is conserved in inelastic collision. However, work & energy consideration may be useful to find the stoppage path.

Answer: $L = \frac{(m_1 v_1)^2 + (m_2 v_2)^2}{2\mu(m_1+m_2)^2 g} = 40m$

Solution: Kinetic energy of both cars right after the collision is $K = \frac{P^2}{2(m_1+m_2)}$ here m_1 and m_2 are their respective masses, and P is their toptal momentum which is conserved at the collisions, i.e. $P^2 = P_x^{\ 2} + P_y^{\ 2} = (m_1 v_1)^2 + (m_2 v_2)^2$.
The stoppage distance L can be found by noting that the work by force of friction should bring the kinetic energy to zero:
$\mu(m_1 + m_2)gL = K = \frac{(m_1 v_1)^2 + (m_2 v_2)^2}{2(m_1+m_2)}$. Therefore,

$$L = \frac{(m_1 v_1)^2 + (m_2 v_2)^2}{2\mu(m_1+m_2)^2 g} = 40m$$

10 points: A sedan of mass ***m*** collides with an SUV of mass ***M*** that stopped at an intersection. The collision between the cars is ***neither purely elastic nor inelastic***. The SUV is pushed forward by distance ***D***, while the sedan comes to a stop after moving by distance ***d<D*** from the collision point, in the same direction. What was the speed of the sedan prior to the collision? The coefficient of kinetic friction between the rubber tires and the asphalt is μ (both drivers were applying brakes at the moment of the collision).

Hint: Regardless of the type of collision, momentum is conserved. All you need is to find speeds/momenta of both vehicles after the collision.

Answer: $v = \sqrt{2\mu g d}\left(1 + \frac{M}{m}\sqrt{D/d}\right)$

Solution: Similar to the 5 pt problem, we can relate stoppage distance to speed after the collision:

$$v_1^2 = 2\mu g d \text{ and } v_2^2 = 2\mu g D$$

The initial speed of the sedan can be found by using the conservation of momentum:

$$v = \frac{mv_1 + Mv_2}{m} = \sqrt{2\mu g d}\left(1 + \frac{M}{m}\sqrt{D/d}\right)$$

CHEMISTRY

This month, the topic is: Dynamic equilibrium in Chemistry.
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 wants to solve Chemistry PoMs, although it is not supposed to provide direct hints.
This month's lecture takes place at 10:30 am on Sunday, September 27. The Zoom invitation is below:

https://us02web.zoom.us/j/4817690592?pwd=T2djSjRETEpDSHFZdWJpYlBTYzdjQT09
Meeting ID: 481 769 0592
Passcode: 879615

The recording is available in our YouTube channel: https://youtu.be/DAfGczRsgrs

During and after the lecture, you will have an opportunity to ask general questions on the topic. The lecture will be recorded, and it can be found at our youtube channel.

In addition, the solutions will be announced via Zoom too, so you will be able to ask questions if you want. The exact date will be announced later. Check our website for that.

5 points:

When you add a powdered soluble substance, e.g. table salt, to water, the solid dissolves, but if you add too much of it, a saturated solution forms, and the rest of the solid does not dissolve anymore. If you leave that mixture in a closed vessel for several weeks, you will see that, although the total amount of the solid hasn't changed, there is a significant change in the particles' appearance: instead of a large number of small grains, one or several large crystals form. Why does it happen?

Hint:

Actually, the hint can be found in the September PoM lecture, which is available here:
https://drive.google.com/file/d/1-wlgrcUTrFdGRRfqZjltCUVt6nVzXb-u/view?usp=sharing
If you cannot open it, please, send a request to Mark Lukin (mark.lukin@stonybrook.edu). The part where Mark tells about evaporation from concave and convex surfaces is the most relevant to the problem.

Answer:

When some solid and its saturated solution are at equilibrium, this state is called a “dynamic equilibrium”, because the apparent cessation is an equilibrium between two processes, dissolution and precipitation. These processes never stop, but, since their speed is equal, we see no changes.

This simple picture works fine when the surface of the solid is flat. When we consider actual solids, the situation is more complex. Indeed, the rate of dissolution depends on how strongly the molecules of the solid are being held by its neighbours. This parameter depends on the number of neighbours. Obviously, the molecules situated at the very tip of the crystal’s vertex are surrounded by a smaller number of neighbours, so they dissolve faster. The molecules (or ions, for ionic compounds) situated in the middle of crystal’s faces are surrounded by a bigger number of neighbours, so they are less likely to escape from the crystal to the solution, and, accordingly, they are more likely to adsorb molecules (ions) from the solution.

Each crystal, independent of their size, has the same number of vertices, but big crystals have bigger faces. That means dissolution from faces does not depend on the crystal’s size, but precipitation (crystallization), which occurs mostly at the plane surface of crystal faces, is faster for bigger crystals.

That means that, whereas the overall dissolution and crystallization rates are equal at equilibrium, a *local* dissolution rate is slightly higher for smaller crystals, and slightly lower for bigger crystals. Accordingly, a *local* crystallization rate is higher for larger crystals and lower for smaller crystals,
so big crystals will be growing bigger and bigger, whereas smaller crystals will become smaller and smaller, and finally disappear.

10 points:

Hardness of fresh or tap water is a property that has no relation to real hardness. It indicates the amount of dissolved salts, usually hydrocarbons and sulfates of calcium and magnesium. When tap water is boiled or heated, calcium salts may precipitate in a form of calcium carbonate ($CaCO_3$) or, more rarely, calcium sulfate ($CaSO_4$). That results in formation of solid deposits on the inner surface of teapots, boilers, and other equipment. For removal of carbonate deposits, a treatment with some acid, such as acetic acid, dilute hydrochloric acid, or citric acid, can be helpful, because solid calcium carbonate participates in an exchange reaction:

$CaCO_3 + 2HCl \rightleftharpoons CaCl_2 + H_2CO_3$

One of the products, carbonic acid, is unstable, it decomposes into water and carbon dioxide (a gas), which escapes to the atmosphere, so the equilibrium shifts to the right side, and all insoluble calcium carbonate converts to the soluble calcium chloride ($CaCl_2$).
That approach does not work for calcium sulfate deposits. Sulfuric acid is not volatile, so if you add HCl to calcium sulfate:

$$CaSO_4 + 2HCl \rightleftarrows CaCl_2 + H_2SO_4$$

sulfuric acid stays in the solution, and the equilibrium remains shifted to the left side, so no conversion of low soluble calcium sulfate into soluble calcium chloride occurs.

Propose the process that would allow conversion of calcium sulfate deposits into something soluble. How many steps are needed for that?

Hint:

Actually, the hint can be found in the September PoM lecture, which is available here:
https://drive.google.com/file/d/1-wlgrcUTrFdGRRfqZjltCUVt6nVzXb-u/view?usp=sharing
If you cannot open it, please, send a request to Mark Lukin (mark.lukin@stonybrook.edu). The part where Mark tells about a solubility product is the most relevant to the problem.

Answer:

Calcium sulfate, as well as any ionic solid (a.k.a. salt), partially dissolves in water, and this process is accompanied by its dissociation.

$$CaSO_4 \rightleftarrows Ca^{2+} + SO_4^{2-}$$

At equilibrium, the concentration of calcium sulfate ions is described by the parameter called *solubility product,* which is equal to

$$SP = [Ca^{2+}][SO_4^{2-}]$$

In this formula, square brackets denote the *actual concentration* of each ion (the square bracket notation is very common in chemical kinetics and thermodynamics, so it is useful to remember it).
From this formula, it is easy to see that the higher SP is, the greater the solid's solubility. For calcium sulfate, SP is 4.93×10^{-5} mol^2L^{-2}.
Another solid, calcium carbonate, is less soluble, and its SP is 3.3×10^{-9} mol^2L^{-2}.

Consider a situation when we take a solid calcium sulfate, add water, and wait until an equilibrium is achieved. At equilibrium, the concentration of calcium ions will be equal to the square root of calcium sulfate's SP, i.e. 7×10^{-3} M. Now, let's add add some amount of carbonate

ions, for example, 1 M of calcium carbonate ($CaCO_3$). This solid dissociates onto calcium ions and carbonate ions, so the concentration of carbonate ions becomes 1M. Taking into account that calcium carbonate (chalk, or limestone) has low solubility (its SP is 3.3×10^{-9} mol^2L^{-2}), calcium ions will interact with carbonate ions, and calcium carbonate will start to precipitate from the solution:

$$Ca^{2+} + CO_3^{2-} \rightleftarrows CaCO_3$$

How will the concentration of calcium ions drop in that process? Taking into account that SP is a product of $[Ca^{2+}]$ and $[CO_3^{2-}]$, and that $[CO_3^{2-}]$ is equal to 1M, the concentration of calcium ions becomes:

$$[Ca^{2+}] = SP_{CaCO3}/[CO_3^{2-}] = 3.3\times10^{-9}/1 = 3.3\times10^{-9}$$

Keeping in mind that the solid calcium sulfate is still present, and it is supposed to be at equilibrium with the solution, it continues to dissolve, but that dissolution process is not compensated by precipitation of $CaSO_4$, because concentration of calcium ion dropped dramatically (now it is 3.3×10^{-9}, not 7×10^{-3}). That means, in the presence of high concentration of carbonate ions, the process of precipitation of the solid calcium sulfate is incapable of compensating its dissolution, so $CaSO_4$ starts to dissolve, and $CaCO_3$ precipitates instead.
Similar to the 5pt problem, this process will last until all $CaSO_4$ is converted into $CaCO_3$ (of course, if a large excess of carbonate ions, for example, high concentration of sodium, potassium or ammonium carbonate, is taken).

What practical consequence all of that has? In contrast to calcium sulfate, calcium carbonate reacts with acids (such as HCl), and soluble calcium salts form. That allows us to easily remove calcium sulfate depositions by treatment of them with a solution of some soluble carbonate, rinsing the newly formed solid with water, and treating it with some acid.

Next Sunday, Oct 18, at 10:30, more detailed explanation will be provided during the Zoom conference:
https://us02web.zoom.us/j/4817690592?pwd=T2djSjRETEpDSHFZdWJpYlBTYzdjQT09
Meeting ID: 481 769 0592
Passcode: 879615

During that Zoom conference, you will be able to ask your questions and discuss your own solutions.

BIOLOGY

5 points:

Groups of organisms often possess what is known as “collective intelligence,” in which flocks and swarms are able to make optimal “decisions” as a group that far exceed the capabilities of any one of its members. Assuming that ants are identical, how do they manage collective transport of objects several times their sizes--so large, in fact, that most ants are incapable of seeing where they’re going? How would behavior change as a function of the size of the swarm?

Answer:

There are two types of ants: those at the head of the motion who can see, and the rest who cannot see. The ones who can’t see react by reinforcing the push and pull of his neighbors, “going with the flow,” and therefore acting dependently. The more ants in the group, the stronger the signal, and therefore the more cooperatively they move--thus small groups of ants transport in a “disordered phase” way while large groups of ants move in an “ordered phase.” The very few number of ants at the head of the motion steer, and thus provide hugely disproportionate influence on the group. The group cycles, with the ones at the front moving to the back, and so forth, so what matters is not the leadership of the specific ant, per se, but rather his role within the geometry of the transport.

FROM: https://www.nature.com/articles/s41567-018-0107-y

“Anyone who has moved furniture together with friends will appreciate that cooperative transport requires some non-trivial communication. Yet ants are adept at collectively moving objects several times their size. How they do so has long been a subject of research, but recent advances have suggested that this communication occurs through the forces the ants exert on the load.”

“Ants who take part in collective transport participate in a large-scale process whereby their individual actions act to coordinate them into a single, cooperative entity. This coordination is achieved via physical interactions in which a carrier ant senses the force generated by the entire carrying group and reacts by aligning its pull with this collective ‘opinion’.”

“The carrying ant groups exhibit the necessary ingredients to support a phase transition in a many-body system. Importantly, group size serves as a control parameter that transitions the group between different phases of motion. Small carrying teams display a disordered phase,

which is characterized by uncoordinated tug-of-war33, whereas large ant groups coordinate into an ordered phase, characterized by more ballistic motion."

"The collective behaviour of ants is strongly dictated by physical principles in which ants play the role of simple coupled particles. Interestingly, this coupled system is maximally responsive to the complex decisions and information provided by few navigationally competent individuals."

10 points:

In May 2020, the Environmental Protection Agency EPA approved release of 750 million genetically engineered mosquitoes in Florida Keys. The mutant mosquito are expected to mate with local, non-mutant mosquitoes to produce nonviable female offspring, which is expected to cause nearly complete eradication of the mosquito population.

- What was the reason for introducing the mutation that kills only females, but not males?

- In which chromosome such lethal mutation should be placed to produce a desired effect, and how all of that works?

- What potential effects (positive and negative) eradication of mosquitoes will have on the ecosystem? Which species of the Florida Keys ecosystem will be most endangered ?

Answer:

Obviously, genetically engineered mosquitoes cannot literally "kill" other mosquitoes, they can, potentially, make the whole population non-viable. That can be achieved by introducing some gene into the population that would decrease the overall survival rate of mosquitoes. The main technical problem is that any gene that decreases the survival rate is usually removed from the population, and after several generations, the population will be dominated by non non-modified organisms, so the effect of introduction of genetically modified mosquitoes vanishes in the next generation. One possibility to avoid that would be to introduce a female-specific lethal gene into the population. If the genetic defect is introduced into the male's genome in such a way that it does not affect viability of male offspring, but female offspring are non-viable, then mutant males are not removed from the population. These males remain viable and fertile; however, when mated to nontransgenic (wild) females, they produce no fertile female. After several generations, all females become eliminated from the population, and all males carry the

same genetic defect. Since the modified gene is not removed from the population (genetically modified males are as viable as non-modified ones), this process will last until all mosquitoes are eradicated.

Where should that genetic modification be introduced? Usually, when we speak about some genetic defect linked to a sex, the first idea is that that defect is allosomal (i.e. is located in a sex chromsome). The problem is that that would work for males, who always inherit their Y-chromosome from the father. In contrast, females have to X-chromosomes, one is matertal, another is fathernal. That means such an approach would not work in that case.

A solution is to introduce dominant homozygous modification of some gene that causes female specific lethality (“dominant homozygous” means that both copies of that gene have the same defect, and the genes are dominant, so the progeny will have the mutant phenotype). For example, there is a specific element of the gene expression system called Yp3fat-body enhancer, which activates some genes in female larvae and adults, but not in males. If a cytotoxic gene is added to one of these genes, that would kill females, but not males. The latter will be totally healthy, and capable of transferring their genetic defect to the next generation of mosquitoes.

Main problem with these genetically modified mosquitoes is that the risks have not been adequately assessed. Potential effects on insect eating birds, or fish eating mosquitoes’ larvae depend on flexibility of food chains in the ecosystem, which are not fully studied. Arguably, species that are closely related to mosquitoes are at greater risk, because if a full eradication of mosquitoes will not be achieved (it was reported that in reality about 3% of female mosquitoes carrying the female specific mutation are still capable of surviving), the mutant gene will continue to exist in the ecosystem and can be transferred to other species, which may lead to unpredictable consequences.

LINGUISTICS

5 points:

Some of the following words from a language spoken in Siberia are translated below (in a random order).

vörny, vörz'yny, vörz'ödny, vörödyštny, vörödny, padmyny, padmödny, lebz'yny, lebny, gazhödyštny, gazhodny, seiny, seiyštny

to move oneself, to make someone late, to eat a little bit, to move someone, to be late, to move someone a little, to have fun, to start moving oneself, to start flying

Determine which translation corresponds to which word and translate the rest of the words. Explain how you reached your conclusion.

Hint:

Solution and Answer:

1. All given words end in *-ny*, so we can assume that this is a verbal infinitive ending.
2. There are 5 different stems: *vör, padmy, leb, gazh, sei*.
3. There are 3 suffixes (other than *-ny*): *-öd, -z'y, -yšt*, distributed as shown in the table below:

-ny	***-z'y-ny***	***-öd-ny***	***-yšt-ny***	***-z'-öd-ny***	***-öd-yšt-ny***
vörny	*vörz'yny*	*vörödny*		*vörz'ödny*	*vörödyštny*
padmyny		*padmödny*			
lebny	*lebz'yny*				
		gazhodny			*gazhödyštny*
seiny			*seiyštny*		

4. There seems to be 4 different meanings that can be combined with each other: basic meaning (to do something oneself, to make someone do something, to start doing something, to do something a little). The table below summarizes the data:

“oneself”	**“make someone”**	**“start”**	**“a little”**	**“make someone”+”a little”**
move oneself	*move someone*	*start moving oneself*		*move someone a little*
be late	*make someone late*			
	have fun			
		start flying		
			eat a little bit	

5. From this table, it is easy to deduce that “move oneself” corresponds to *vörny* (since it’s the only row with 4 or more words; now, *vörz’yny* and *vörödny* correspond to “move someone” and “start moving”.
6. If we assume that *vörz’yny* is “to move someone” (and *-z’y* corresponds to “make someone”), then “to move someone a little” would be *vörz’ödny*, so *-öd* means “a little”, and we are missing the translation for “to start moving oneself” -- contradiction.
7. Therefore, *vörödny* is “to move someone” (and *-öd* corresponds to “make someone”), *vörz’yny* is “to start moving” (and *-z’y* corresponds to “start”), *vörz’ödny* is “to start moving someone”, and *vörödyštny* is “to move someone a little” (and *-yšt* means “a little”)
8. Now, *padmyny* and *padmödny* correspond to “be late” and “make someone late”. “To have fun” is *gazhodny*, “to eat a little” is *seiyštny*, “to start flying” is *lebz’yny*. The rest of the words can be translated as follows: *lebny* is “to fly”, *seiny* is “to eat”, *gazhödyštny* is “make someone have fun a little”.

10 points:

The phrases below from a language spoken in southeast Asia have the given translations in random order.

sampeyan murid adhi aku
murid adhi kanca sampeyan
adhi aku kanca murid sampeyan
aku murid kanca sampeyan
adhi aku murid sampeyan

My younger brother is a friend of your student.

You are the student of my younger brother.
the student of the younger brother of your friend
My younger brother is your student.
I'm the student of your friend.

a. Establish the translations of the words and describe the structure of such sentences from this language. Explain how you reached your answer.
b. According to what you discover, what is another possible translation of the 2nd phrase?

Solution and Answer:

1. Notice that there are only 5 different words used in the original data, so it would be reasonable to assume that English words like "you, your" and "I, my" correspond to just one word in this language. We should also assume that "younger brother" corresponds to just one word. Hence, there are 5 different meanings: (1) I, me, my (2) you, your (3) younger brother (4) friend (5) student.
2. Let's create a table for each of the data sentences and each of the translations to indicate which words are used where.

	1-data	**2-data**	**3-data**	**4-data**	**5-data**
sampeyan	+	+	+	+	+
murid	+	+	+	+	+
adhi	+	+	+		+
aku	+		+	+	+
kanca		+	+	+	

	1-translation	**2-translation**	**3-translation**	**4-translation**	**5-translation**
I/my	+	+		+	+
you/your	+	+	+	+	+
younger brother	+	+	+	+	
student	+	+	+	+	+
friend	+		+		+

3. The word *kanca* means “friend” since they both occurs in 3 sentences. The words *sampeyan* and *murid* correspond to “you/your” and “student”, but so far it is unclear which corresponds to which. Therefore, *adhi* and *aku* correspond to “I/my” and “younger brother” (again, unclear which to which).
4. Sentence 3-data has 5 words, and so does 1-translation, so that’s the correct correspondence.
5. Sentences 1-data and 5-data have the same 4 words, and so do 2-translation and 4-translation (but which corresponds to which is still unclear).
6. Based on that, we have to use guess and check to figure out the right translations. The only consistent rules can be stated if: 1-data=2-translation, 5-data=4-translation, 2-data=3-translation, 4-data=5-translation, 3-data=1-translation; and *sampeyan*=”you/your”, *murid*=”student”, *adhi*=”younger brother”, *aku*=”I/my”, *kanca*=”friend”.
7. The order of words is the following: Subject-Object-Verb; the Possessive follows the Noun.
8. The 2nd sentence is *murid adhi kanca sampeyan*, which has the following word-for-word translation: “student younger_brother friend your”, and it can have another translation: “student is the younger brother of your friend”.

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:

Mark decided to digitize his library. He has ***n*** books on his shelf. ***k*** counselors volunteered to help him scan all the books. (You can presume that k <= n). Mark is very particular in the order the books appear on the shelf at any given moment, and therefore counselors are instructed not to mix or move the books, and each counselor can only scan a set of consecutive books.

Your program should read the input file **input.txt**, which consists of 2 rows. The first row contains space-separated values of *n* and *k*. The second row contains n space-separated integers representing the size of each of the books (i.e., number of pages). Example input file:

```
10 3
50 100 75 66 350 254 39 111 205 321
```

You can assume that all books have a positive integer number of pages.

5 points:

Your program should calculate the minimum and maximum number of pages a counselor may need to scan considering all possible divisions of labor.

The program should produce output file **output.txt**, which consists of 2 space-separated numbers - minimum and maximum. For example above, the file will contain:

```
39 1421
```

Solution:

Java:

```
/*
if k == 1 then min and max are the entire collection of the books
if k == 2 then
```

```
  min is either the 1st or last book
  max is all the books minus either the 1st or last book
else
  The minimum number of pages is the number of pages in a single smallest book because we can
  isolate the book with partitions on the left and right of it.
  The maximum number of pages is the max sum of pages in n-(k-1)=n-k+1 consecutive books, i.e.
  when k-1 counselors take only 1 book each and the last counselor takes all the rest.
We will calculate these 2 number "on the fly" while scanning the books array once, i.e.
the complexity is O(n).
*/

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.Arrays;

public class Books5 {
  private int[] pages;
  private int k;
  private int mini;
  private int maxi;

  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();
    // also verify that they all > 0
    if(!Arrays.stream(numbers).allMatch(i -> i > 0))
      throw new AssertionError("all numbers must be positive");
    if(numbers.length != 2)
      throw new Exception("1st line must have 2 integers");
    int n = numbers[0];
    k = numbers[1];
    if(k > n)
      throw new Exception("k > n");

    line = reader.readLine().trim();
    nums = line.split("\\s*[\\s,]\\s*");
    if(nums.length != n)
      throw new Exception("2nd line must have n integers");
    pages = Arrays.stream(nums).map(s ->
       Integer.valueOf(s)).mapToInt(Integer::intValue).toArray();
    if(!Arrays.stream(pages).allMatch(i -> i > 0))
      throw new AssertionError("all numbers must be positive");
  }

  void calc() {
    if(k == 1) { // a lonely counselor gets all the work
      mini = Arrays.stream(pages).sum();
      maxi = mini;
    }
    else if(k == 2) {
      mini = Math.min(pages[0], pages[pages.length-1]);
      maxi = Arrays.stream(pages).sum();
```

```
      maxi = Math.max(maxi-pages[0], maxi-pages[pages.length-1]);
    }
    else {
      int n = pages.length;
      mini = Integer.MAX_VALUE;
      maxi = 0;
      int m = 0; // number of books in "maxi"
      int s = 0; // running sum
      for(int i=0; i<n; i++) { // go book by book
        int cur_pages = pages[i];

        if(cur_pages < mini)
          mini = cur_pages;

        if(m < n-k+1) {
          m++;
          maxi += cur_pages;
          s = maxi;
        }
        else {
          s = s - pages[i-m] + cur_pages; // by this time m = n-k+1
          if(s > maxi)
            maxi = s;
        }
      }
    }
  }

  void output() throws Exception {
    System.out.print(String.format("%d %d\n", mini, maxi));
    try(FileWriter out = new FileWriter("output.txt")) {
      out.write(String.format("%d %d\n", mini, maxi));
    }
  }

  public static void main(String[] args) throws Exception {
    Books5 books = new Books5();
    books.input();
    System.out.println(Arrays.toString(books.pages));
    books.calc();
    books.output();
    System.out.println("end.");
  }
}
```

Python:

```
"""
if k == 1 then min and max are the entire collection of the books
if k == 2 then
  min is either the 1st or last book
  max is all the books minus either the 1st or last book
else
  The minimum number of pages is the number of pages in a single smallest book because we can
  isolate the book with partitions on the left and right of it.
  The maximum number of pages is the max sum of pages in n-(k-1)=n-k+1 consecutive books, i.e.
  when k-1 counselors take only 1 book each and the last counselor takes all the rest.
We will calculate these 2 number "on the fly" while scanning the books array once, i.e.
```

```
the complexity is O(n).
"""

import re

# read and parse input file
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) != 2:
    raise Exception("first line must have 2 numbers")
  n = numbers[0]
  k = numbers[1]
  if n<=0 or k<=0 or k>n:
    raise Exception("invalid input")

  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
  pages = [int(x) for x in numbers_str]
  if len(pages) != n:
    raise Exception("number of books must be n")
  if len([x for x in pages if x<=0]) > 0:
    raise Exception("number of pages must be > 0 in a book")
print(pages)

if k == 1: # a lonely counselor gets all the work
  mini = sum(pages)
  maxi = mini
elif k == 2:
  mini = min(pages[0], pages[-1])
  maxi = max(sum(pages[1:]), sum(pages[0:-1]))
else:
  mini = float('inf')
  maxi = 0
  m = 0 # number of books in "maxi"
  s = 0 # running sum
  for i in range(len(pages)): # go book by book
    cur_pages = pages[i]

    if cur_pages < mini:
      mini = cur_pages

    if m < n-k+1:
      m += 1
      maxi += cur_pages
      s = maxi
    else:
      s = s - pages[i-m] + cur_pages # by this time m = n-k+1
      if s > maxi:
        maxi = s

print(f"{mini} {maxi}")

with open("output.txt", "w") as out_file:
```

```
  out_file.writelines(f"{mini} {maxi}\n")

print("end.")
```

10 points:

Your program should find the ***fairest*** division of labor (partition of the shelf into k sets of consecutive books). Fairest is defined as a division with the minimum of the total number of pages the busiest counselor (i.e. the counselor who will have to scan the most of the pages) will have to scan.

The program should produce output file **output.txt**, which consists of k lines. Each line *i* will start with the number of pages the *i*-th counselor has to scan, followed by a space, dash, space and then space-separated list of book sizes comprising that total.
For example, for an input file:

```
6 2
10 20 30 40 50 60
```

the output file should contain:

```
100 - 10 20 30 40
110 - 50 60
```

Solution:

Java:

```
/*
The explanation is given by
Steven Skiena "The Linear Partition Problem"
https://www3.cs.stonybrook.edu/~algorith/video-lectures/1997/lecture11.pdf
*/

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;

public class Books10 {
  static int INF = Integer.MAX_VALUE;
  private int[] pages;
  private int k;

  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();
  // also verify that they all > O
  if(!Arrays.stream(numbers).allMatch(i -> i > 0))
    throw new AssertionError("all numbers must be positive");
  if(numbers.length != 2)
    throw new Exception("1st line must have 2 integers");
  int n = numbers[0];
  k = numbers[1];
  if(k > n)
    throw new Exception("k > n");

  line = reader.readLine().trim();
  nums = line.split("\\s*[\\s,]\\s*");
  if(nums.length != n)
    throw new Exception("2nd line must have n integers");
  pages = Arrays.stream(nums).map(s ->
     Integer.valueOf(s)).mapToInt(Integer::intValue).toArray();
  if(!Arrays.stream(pages).allMatch(i -> i > 0))
    throw new AssertionError("all numbers must be positive");
}

static int[][] linearPartition(int[] a, int k) {
  int n = a.length;
  int[][] table = new int[n][k];
  int[][] solution = new int[n-1][k-1];

  // initialize
  table[0][0] = a[0];
  // compute prefix sums
  for(int i=1; i<n; i++) {
    table[i][0] = a[i] + table[i-1][0];
  }
  // initialize boundary condition
  for(int j=1; j<k; j++) {
    table[0][j] = a[0];
  }

  // fill the rest
  for(int i=1; i<n; i++) {
    for(int j=1; j<k; j++) {
      int current_min = -1;
      int minx = INF;
      for(int x=0; x<i; x++) {
        int s = Math.max(table[x][j-1], table[i][0] - table[x][0]);
        if(current_min < 0 || s < current_min) {
          current_min = s;
          minx = x;
        }
      }
      table[i][j] = current_min;
      solution[i-1][j-1] = minx;
    }
  }
```

```
    return solution;
  }

  static List<List<Integer>> reconstructPartition(int[] a, int[][] solution, int k) {
    List<List<Integer>> result = new ArrayList<>();
    int n = solution.length;
    k = k - 2;
    while(k >= 0) {
      List<Integer> inner = new ArrayList<>();
      for(int i=solution[n-1][k]+1; i<n+1; i++) {
        inner.add(a[i]);
      }
      result.add(inner);
      n = solution[n-1][k];
      k--;
    }

    List<Integer> inner = new ArrayList<>();
    for(int i=0; i<n+1; i++) {
      inner.add(a[i]);
    }
    result.add(inner);
    Collections.reverse(result);
    return result;
  }

  void output(List<List<Integer>> res) throws Exception {
    try(FileWriter out = new FileWriter("output.txt")) {
      for(List<Integer> row : res) {
        out.write(String.format("%d - %s\n", row.stream().mapToInt(i -> i.intValue()).sum(),
                  row.stream().map(String::valueOf).collect(Collectors.joining(" "))));
      }
    }
  }

  public static void main(String[] args) throws Exception {
    Books10 books = new Books10();
    books.input();
    System.out.println(Arrays.toString(books.pages));
    int[][] solution = linearPartition(books.pages, books.k);
    List<List<Integer>> res = reconstructPartition(books.pages, solution, books.k);
    System.out.println(res);
    books.output(res);
    System.out.println("end.");
  }
}
```

Python:

```
"""
The explanation is given by
Steven Skiena "The Linear Partition Problem"
https://www3.cs.stonybrook.edu/~algorith/video-lectures/1997/lecture11.pdf
"""

import re
from operator import itemgetter
```

```
# read and parse input file
def input():
  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) != 2:
      raise Exception("first line must have 2 numbers")
    n = numbers[0]
    k = numbers[1]
    if n<=0 or k<=0 or k>n:
      raise Exception("invalid input")

    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
    pages = [int(x) for x in numbers_str]
    if len(pages) != n:
      raise Exception("number of books must be n")
    if len([x for x in pages if x<=0]) > 0:
      raise Exception("number of pages must be > 0 in a book")
    return (pages, k)

def linear_partition(a, k):
  if k <= 0:
    return []
  n = len(a) - 1
  if k > n:
    return map(lambda x: [x], a)
  table, solution = linear_partition_table(a, k)
  k, ans = k-2, []
  while k >= 0:
    ans = [[a[i] for i in range(solution[n-1][k]+1, n+1)]] + ans
    n, k = solution[n-1][k], k-1
  return [[a[i] for i in range(0, n+1)]] + ans

def linear_partition_table(a, k):
  n = len(a)
  table = [[0] * k for x in range(n)]
  solution = [[0] * (k-1) for x in range(n-1)]
  for i in range(n):
    table[i][0] = a[i] + (table[i-1][0] if i else 0) # prefix sums
  for j in range(k):
    table[0][j] = a[0] # boundary condition
  for i in range(1, n):
    for j in range(1, k):
      table[i][j], solution[i-1][j-1] = min(((max(table[x][j-1], table[i][0]-table[x][0]), x)
for x in range(i)), key=itemgetter(0))
  return (table, solution)

def output(a):
  with open("output.txt", "w") as out_file:
    for i in range(len(a)):
      out_file.write(f"{sum(a[i])} - {' '.join(str(x) for x in a[i])}\n")

pages, k = input()
```

```
print(pages)

a = linear_partition(pages, k)
print(a)

output(a)
print("end.")
```