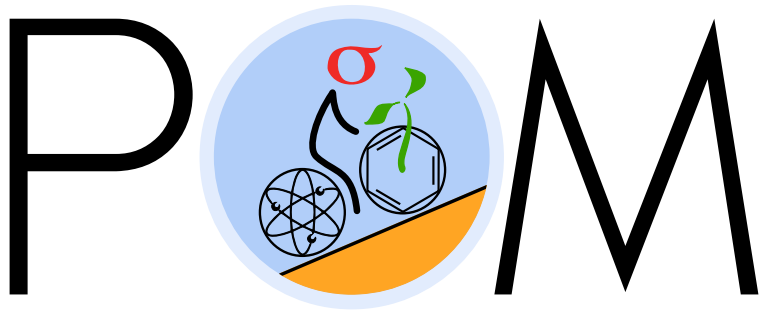


SigmaCamp's *Problem of the Month* Contest

OCTOBER 2025

IMPORTANT: For the 2025-2026 season, POM is experimenting with two format changes:

- **Video solutions** must be submitted for each problem.
- Each month will include a **project** (worth 30 points) in place of one or two subjects.
- We now have monthly **POM office hours** where you can ask questions about this month's POM problems! See sigmacamp.org/ pom/office-hours for details.

Video Submissions

Starting this year, **all solutions must be accompanied by short videos explaining your work.**

- Videos must be at most **2 minutes long** for 5pt/10pt problems, and at most **6 minutes long** for projects.
- **Solutions must be narrated**, but you do not need to show your face. Acceptable formats include:
  - Screensharing slides or a drawing app (e.g., MS Paint) with narration.
  - Recording a whiteboard, paper, or easel with narration (contents may be pre-written).
  - Speaking directly to the camera.
- **Submit videos as links** (Google Drive, Youtube, Dropbox, etc.). Extra requested files may be submitted as a single PDF file per problem.
- For coding problems, submit your code along with a video explaining your submission. Only **Python-3** (`.py`) and **Java** (`.java`) code submissions are accepted.

Please see sigmacamp.org/pom for full details on the 2025 POM format change.

Project (30 points)

Biology and Chemistry

Introduction

In the history of scientific experimentation, some of the most transformative discoveries arose from experiments whose designs were strikingly simple yet methodologically rigorous. Two well-known examples are Gregor Mendel's pea plant crosses and Robert Boyle's investigation of gas pressure and volume.

Mendel used garden peas to study inheritance. By isolating traits such as seed shape or flower color, he controlled fertilization between true-breeding plants and meticulously counted the traits of thousands of offspring across generations. What made his work rigorous was not the complexity of tools, but the precision in experimental design. From this, Mendel derived statistical ratios that revealed the particulate nature of heredity, earning him the title of the "father of genetics."

Similarly, Boyle's 17th-century gas experiments relied on a J-shaped glass tube and mercury. He trapped a pocket of air at one end and varied the pressure on it by adding mercury, while measuring the corresponding volume change. Through repeated trials, Boyle demonstrated a precise mathematical relationship: pressure multiplied by volume is constant when temperature is held steady. This conclusion, known as Boyle's Law, emerged from a methodology built on simple apparatus, systematic variation, and quantitative recording.

Often, experiments are designed using the "scientific method." The scientific method is defined as a "dynamic process that involves objectively investigating questions through observation and experimentation."[1] Additional resources about the scientific method can be found here.

Instructions

There is an experimental and theoretical section to this project. The experimental section is worth 18 points, and the theoretical section is worth 12 points. There are opportunities for partial credit in all parts of the project.

The solution will be graded based on the quality of the experiment, experimental design, and the quality of the work itself.

If your experiment fails, do not worry. There is still an opportunity to receive points. Refer to the section after the theoretical questions for questions about your experiment.

Experimental Portion

In this project, you will design and conduct a controlled investigation to test how effectively yeast produces carbon dioxide when provided with different beverages as its primary sugar source. Your task is to test at least three different liquids of your choice and determine which one allows yeast to ferment most effectively. You are required to collect quantitative data and establish a result based on your data.

You are encouraged to be creative in how you design and conduct a controlled investigation to test how effectively yeast produces CO_2 during the fermentation of different beverages as its only sugar source. We will be looking for scientific rigor, optimization, and implementation. We encourage using modern scientific approaches based on realistic experimental designs.

[1] https://extension.unr.edu/publication.aspx?PubID=4239

Theoretical Questions

Each question below is worth 3 points.

1. Different drinks with the same sugar content could produce different volumes of CO_2. If that occurs, how could you explain this?
2. What are the ideal parameters for fermentation, and how do sugary carbonated beverages deviate from them?
3. Equate the chemical reaction occurring.
4. Among the processes of glycolysis, fermentation, and respiration, which results in the greatest change in volume, and why?

If your experiment fails

There is still the opportunity to receive points, even if something goes wrong with your experiment. Recall that a significant part of scientific experiments fail, and drawing correct conclusions from the experiments that went wrong is a part of a scientist's job. If your experiment fails, analyze possible reasons and, if you can, propose solutions: you may also get some points for this. Your analysis may include the answers to the following questions:

1. What could you have done to improve your experiment?
2. Describe what you did and some reasons why it might have failed.
3. If you decide to redesign your experiment at any time, please submit all your results and describe what you changed.

Hint:

This project should be carried out in the same way as any scientific research is done. One important feature of scientific research is that the scientist simultaneously acts as their own devil's advocate. The primary role of the devil's advocate is questioning every observation and conclusion you make, and asking: "What if your observations can be explained by some different reasons, which are irrelevant to the phenomenon you are trying to measure?"

Solution:

Featured Solution
by Nathaniel Hunter
https://www.youtube.com/watch?v=mIqjXby17kE

Featured Solution
by Masha Shablygin
https://www.youtube.com/watch?v=EmEBRiKG81s

Featured Solution

by Lili Sigalov

https://www.youtube.com/watch?v=CPljefPH73A

Official solution below.

Experimental Solution

Due to the creative nature of this problem, there is not one distinct solution. Rather, there are a few aspects of experimental design we were looking for among submissions.

Experiments are often done in replicates or triplicates to account for confounding variables. If you had repeated your experiment multiple times, this increased your score.

Additionally, controls are one of the most important aspects of experimental design. Having a positive and a negative control are crucial to be able to validate your results. To read more about positive and negative controls and their differences, click here.

If you did not know what to use as your controls, this is a good place to look into the literature. When thinking of scientific experiments, scientists are looking for the gap in the literature, or questions that have yet to be asked or answered.

Possible experimental setup: inverted graduated cylinder full of water or party balloons, connected to the end of the fermenting flask. Alternate setup: fermenting on an open flask, measuring mass loss (departing CO2) over time with a scale.

Point Distribution

- Experimental Design (4 points for a way to capture the gas or measure the mass loss + 2 points for using consistent volume/mass of beverage to yeast) - 6 points
- Temperature Control - 2 points
- Beverage Choices (rigor, proper design, not water) - 2 points
- Quantitative Data (consistent and accurate results) - 4 points
- Negative Control - 2 points
- Replicates/Triplicates - 2 points
- BONUS POINTS - 2 points for positive control

Beverage constraints: should be reasonably different (not water), possible to ferment, and should differ in one of the following variables:

- different sugar types
- sugar levels
- pH
- food preserving agents
- ABV

Theoretical Questions

Each question below is worth 3 points.

1. Different types of sugar will ferment at different rates. Also, different beverages may have different pH (optimal pH between 4 and 6), which may deactivate the yeast. The presence of other micronutrients in the medium may enhance yeast growth, while high salinity or the presence of food preservatives may inhibit it. (1 pt for each correct explanation)

2. Ideal conditions (0,5 pt for each):

   - **Temperature:** 28°C to 35°C
   - **pH:** Between 4 and 6.
   - **nutrients availability:** Sugars + nitrogen source (aminoacids) + vitamins + microminerals.
   - **Osmotic pressure:** Low salinity level and not too high sugar concentration to avoid yeast plasmolysis.

   Sugary carbonated beverages may deviate because (1 pt for at least 3)

   - High concentration of CO_2 dissolved lowers the pH.
   - the presence of naturally occurring (e.g. citric acid or ascorbic acid) or added acids (e.g. phosphoric acid) may turn the pH too low for yeast growth and fermentation.
   - The medium may be insufficient to provide nutrients to yeast (e.g. lack of amino acids).
   - High osmotic pressure.
   - Presence of food preserver (e.g. sorbate or benzoate)

3. For a generic hexose:
   $C_6H_{12}O_6 \rightarrow 2CO_2 + 2C_2H_5OH$

4. The greatest change in volume is due to fermentation:(1 pt)

   - **Glycolysis**

     $C_6H_{12}O_6(aq)+2ADP(aq)+2P_i(aq)+2NAD^+(aq) \rightarrow 2C_3H_3O_3{}^-(aq)+NADH(aq)+2ATP(aq)+2H^+(aq)+H_2O(l)$

     No gas was produced, therefore no significant change in volume.

   - **Fermentation**

     $$C_6H_{12}O_6(aq) \rightarrow 2CO_2(g) + 2C_2H_5OH(aq)$$

     2 mols of gas per mol of glucose.

   - **Respiration**

     $$C_6H_{12}O_6(aq) + 6O_2(g) \rightarrow 6CO_2(g) + 6H_2O(l)$$

     6 mols of gas per mol of glucose; however, 6 mols of O_2 are also consumed. No significant change in volume.

   (2 pts for equations with explanation)

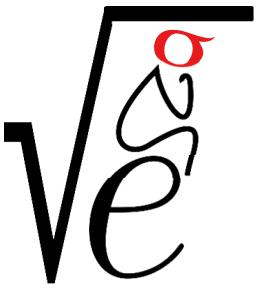
If your experiment fails

(a) What could you have done to improve your experiment? (3 pts)
Consider only specific changes such as replacing the gas collector system, adding replicates or controlling the temperature. Do not consider generic answers like "getting better data points"

(b) Describe what you did and some reasons why it might have failed. (3 pts)
Campers should address problems related to their specific setup or execution, such as leakage or accidental yeast inactivation by overheating.

(c) If you decide to redesign your experiment at any time, please submit all your results and describe what you changed. (3 pts)
Full points if new submission makes sense.

Mathematics



For all mathematics problems, please provide full justification. **Do not include any code** in your submission unless specified otherwise — all code submissions will be awarded no points.

5 points:

The number 20! = 2 432 902 008 176 640 000 has 41040 divisors. How many of them are odd?

Hint:

Consider prime decompositions.

Solution:

Featured Solution
by Valerie Rabinovich
https://www.youtube.com/watch?v=LVl_UxVE4JM

Official solution below.

Answer: 2160

The numbers $1, \ldots, 20$ contain 10 multiples of 2, 5 multiples of 4, 2 multiples of 8 and 1 multiple of 16. Hence, the power of 2 in the prime decomposition of 20! is $10 + 5 + 2 + 1 = 18$ and $20! = 2^{18}p$, where p is an odd number.

If d is a divisor of p, then $d, 2d, \ldots, 2^{18}d$ are all divisors of 20!; all of them except d itself are even. Thus, the 41040 divisors are split into $41040/19 = 2160$ groups, and each group contains exactly one odd divisor, so there are 2160 odd divisors.

You can also produce the full decomposition $2^{18} \times 3^8 \times 5^4 \times 7^2 \times 11 \times 13 \times 17 \times 19$ and count the amount of odd factors as $(8+1)(4+1)(2+1)(1+1)(1+1)(1+1)(1+1) = 9 \times 5 \times 3 \times 2^4 = 2160$.

10 points:

(a) A stash of several identical watermelons was stored at SigmaCamp. At lunch, the SigmaCampers collectively ate 10 of the watermelons. Everyone present received the same amount of watermelon.

At snack time later that day, some campers were still full from lunch, but a team consisting of only 11 of the campers wanted more watermelon. Each of them ate half as much watermelon as they did at lunch. By the end of snack time, all the remaining watermelons were gone. How many watermelons were stored initially?

(b) Write your own problem that uses the same idea as the problem above to create a solvable problem with exactly one solution. Your problem cannot use any of the three numbers from the original problem (10, 11, half). Creative scenarios that do not involve consuming food or drink are encouraged.

Hint:

No hint this month.

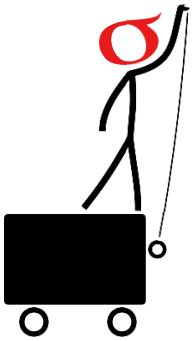
Solution:

Answer: (a) 11 watermelons

If there are w watermelons and c campers originally, then during lunch each camper eats $\frac{10}{c}$ watermelon, and then during snack time all 11 campers eat $11 \cdot \frac{10}{2c}$ watermelon. Thus, $\frac{55}{c} = w - 10$. We know that c and w are whole numbers and that $c > 11$. Then c has to be a factor of 55 and these are $\{1, 5, 11, 55\}$, so the only option is $c = 55$ and then $w = 11$.

The device used to ensure a unique solution is to create a scenario that implies whole numbers, to use numbers that combine to create a number (in this case 55) with very few divisors, and then to add conditions that exclude all divisors except one.

Physics



5 points:

There is a plate attached to the bottom of a long tube by a Hookean spring of spring constant k (if you don't know what this means, have a look at the Additional Information section below). A ball of mass M rests on the plate, as shown in Figure 1A. For this problem you may neglect friction and air resistance, and you may assume the mass of both the plate and the spring to be 0. Let the height of the spring at rest with no mass on it correspond to $y = 0$.

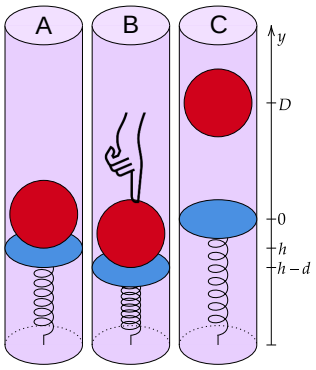


Figure 1: Diagram for the Physics 5 pt problem

(a) (0 points) List the forces on the ball, their magnitudes, and their directions. Draw a free body diagram showing all of them. What is the total force? What condition must hold for the ball to be at rest?

This question is worth 0 points because it is purely here to get you warmed up. The answer is presented at the bottom of the problem, but try to do it yourself before looking.

(b) (1 point) At what height h does the ball rest? What is the gravitational potential energy of the ball relative to $y = 0$? What is the energy stored in the spring? What is the kinetic energy of the ball?

Now imagine that someone comes along and pushes the ball down by an additional distance d. The ball is then released and shoots up into the air.

(c) (3 points) What is the maximum height D the ball reaches?

However, in no real spring does the force perfectly scale linearly with the displacement. Assume instead that

$$F(y) = -ky - ly^2.$$

In this case the potential energy due to this kind of spring is given by the usual quadratic Hookean term, plus a non-Hookean correction:

$$V = \underbrace{\frac{1}{2}ky^2}_{\text{Hookean}} + \underbrace{\frac{1}{3}ly^3}_{\text{non-Hookean}}$$

For realistic springs, l is usually very small and the non-Hookean terms can be ignored.

(d) (1 point) What is the maximum height the ball now reaches?

Answer to (a) There are two forces on the ball: the gravitational force pointing downwards and the spring force pointing up. The magnitude of the spring force will be $F_s = -hk$, by Hooke's Law (remember that h is negative). It will be pointing up. The magnitude of the gravitational force will be $F_g = mg$, and it will be pointing down.

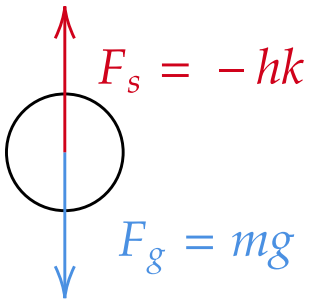


Figure 2: Free body diagram for part (a) of the 5pt physics problem.

The total force is $\vec{F}_{\text{net}} = \vec{F}_s + \vec{F}_g$. The horizontal component will be $F_{\text{net},x} = 0$ and the vertical component will be $F_{\text{net},y} = F_s - F_g = -hk - mg$. In order for the ball to be at rest, the acceleration must be 0, so the net force must also be 0. Thus, $-hk - mg = 0$.

Additional Information. Springs and objects that can be modeled like springs, can be frequently found in many physical systems. The simplest type of spring, called a "Hookian spring", exerts a force if it is stretched or compressed via the following equation

$$F = -k\Delta x,$$

where k is the spring constant and Δx is how much the spring is stretched/compressed[2] from its equilibrium position.

It is often useful to think about these quantities called the kinetic and potential energy of the mass on a spring, denoted by T and V_s respectively:

$$T = \frac{1}{2}mv^2, \quad V_s = \frac{1}{2}k(\Delta x)^2,$$

where m is the mass of the item on spring, v is its speed and Δx is once again how much the spring is stretched from its equilibrium position.

It turns out, that for systems where force only depends on the position of the particle, the total energy of the system is conserved over time[3]:

$$E_{tot} = T + V_{tot}.$$

For example, in the case of a mass on a spring, if at every point in time we measure the speed v, how much the spring is stretched Δx and use it to compute E_{tot} as above, we always get the same number [4].

The potential energy a one particle system has due to gravity is given by

$$V_g = mgh,$$

where m is the mass of the particle, g is the gravitational constant and h is the elevation of the particle above ground (or any fixed reference point). In the case of multiple forces acting on one particle, V_{tot} is the sum of potential energies due to each force involved.

[2] We assume the convention that positive Δx refers to the spring being stretched and negative to the string being compressed by Δx.

[3] You can check this for the spring by considering some small time interval Δt, and comparing the energy of the system at time t with its energy at time Δt. It is useful to make the approximations that $v(t+\Delta t) = v(t)+\Delta t \cdot a(t)$ and $x(t+\Delta t) = x(t)+v(t)\Delta t$ and seeing that the energy essentially does not change over this arbitrarily small time change Δt.

[4] Note that this number is the same as the same system evolves over time. Energy can be different, for a system with a different starting v and Δx.

Hint:

No hint this month.

Solution:

- **Answer to (a)** There are two forces on the ball: the gravitational force pointing downwards and the spring force pointing up. The magnitude of the spring force will be $F_s = -hk$, by Hooke's Law (remember that h is negative). It will be pointing up. The magnitude of the gravitational force will be $F_g = mg$, and it will be pointing down.

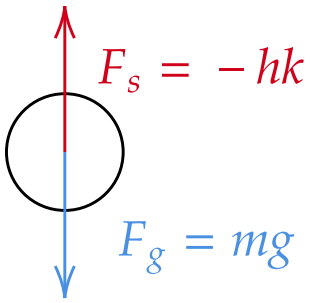


Figure 3: Free body diagram for part (a) of the 5pt physics problem.

The total force is $\vec{F}_{\text{net}} = \vec{F}_s + \vec{F}_g$. The horizontal component will be $F_{\text{net},x} = 0$ and the vertical component will be $F_{\text{net},y} = F_s - F_g = -hk - mg$. In order for the ball to be at rest, the acceleration must be 0, so the net force must also be 0. Thus, $-hk - mg = 0$.

- **Answer to (b)** If the ball is at rest, then the net force acting on the ball must be 0. We therefore get that:

$$\begin{aligned} 0 = F_{net} = F_s + F_g = -kh - mh \\ -kh = mg \\ h = -\frac{mg}{k} \end{aligned} \tag{1}$$

The energy stored in the spring is then :

$$E_s = \frac{1}{2}kh^2 = \frac{1}{2}k\left(\frac{mg}{k}\right)^2 = \frac{m^2g^2}{2k}$$

THe ball also has gravitational potential energy:

$$E_p = mgh = -\frac{m^2g^2}{k}$$

- **Answer to (c)** Initially the ball is held down to a point $h - d$. Since it is stationary, it has 0 kinetic energy. Its total energy is then just the sum of its potential energies:

$$E_{initial} = T + V = 0 + \frac{1}{2}k(h-d)^2 + mg(h-d)$$

Once the ball is released and reaches its maximum height at point D, its speed should be 0 (as otherwise it would keep flying up further). It is no longer attached to the spring and therefore only has gravitational potential energy:

$$E_{final} = mgD$$

By conservation of energy, we get that $E_{initial} = E_{final}$. Substituting the expressions for the energies, we solve for D

$$\frac{1}{2}k(h-d)^2 + mg(h-d) = mgD$$

$$D = (h-d) + \frac{1}{2mg}k(h-d)^2 \tag{2}$$

We can then substitute our expression for h from equation 1 to get D in terms of m, g, k, and d.

$$D = -\left(\frac{mg}{k} + d\right) + \frac{1}{2mg}k\left(\frac{mg}{k} + d\right)^2 = \frac{kd^2}{2mg} - \frac{mg}{2k}$$

- **Answer to (d)** Note that a non=hookean spring has a new force law. We therefore need to recalculate the rest height h away from equilibrium. As before, we want the net force to be 0:

$$0 = F_g + F_s = -kh - lh^2 - mg$$

Using the quadratic formula we get that:

$$h_{nonhookean} = \frac{-k \pm \sqrt{k^2 - 4lmg}}{2l} \tag{3}$$

Note that both of the solutions are negative and therefore below the equilibrium point. We will take the solution with the positive square root, as it will be closer to the equilibrium of the spring (less compressed), as we would express this non-hookean approximation to be valid only for small displacements from equilibrium.

We then once again use conservation of energy to find D

$$\frac{1}{2}k(h-d)^2 + \frac{1}{3}l(h-d)^3 + (h-d) = mgD$$

$$D = \frac{k}{2mg}(h-d)^2 + \frac{l}{3mg}(h-d)^3 + h - d \tag{4}$$

Where h is given by equation 3.

10 points:

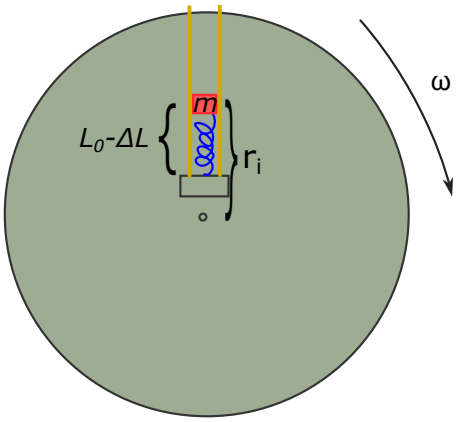
Consider the system shown in Fig. 4, consisting of a merry-go-round with a mass-spring setup. The surface has a static and and kinetic coefficient of friction denoted by μ_s and μ_k, respectively. A mass m is attached to the spring (which has spring constant k) that is anchored to a short vertical wall. The mass can only slide radially along frictionless rails on the merry-go-round.

Initially, the spring is compressed by a distance ΔL relative to its rest length L_0. The only force preventing the spring from uncompressing is the force of static friction on the mass. Taking into account the distance of the wall from the center of the merry-go-round and the spring's compression, the mass starts at an initial radius r_i.

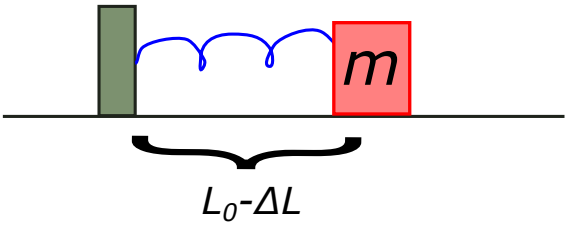
The merry-go-round starts at rest, and gradually accelerates.

Part (a): Find the angular velocity ω the merry-go-round must accelerate to for the static friction to be overcome, causing the mass to start slipping.

Part (b): The merry-go-round is set to spin at the fixed angular velocity ω you solved for in the previous part. As the mass begins to move, the spring decompresses, but the mass remains permanently affixed to the spring's end. Find the maximum radius r_f that the mass reaches.



(a) A top-down view of the system. A mass m is loaded onto a spring of constant k compressed a length ΔL below its rest length L_0. The mass sits at a distance r_i from the center. The orange lines are frictionless rails attached to the merry-go-round that keep the mass only moving in the radial direction.



(b) A side view of the system in figure (a). The system starts off compressed by length ΔL from its equilibrium length. The surface is rough, with a coefficient of static friction μ_s and kinetic friction μ_k

Figure 4

Hint:

Recall that work done by a constant force is

$$W = -F \cdot \Delta x$$

Where Δx is the displacement of the object by force of magnitude F. This equation also assumes that the displacement is along the same direction as the force.
Use this to compute the energy lost by the mass due to kinetic friction, as the mass moves away from its initial position.

Solution:

Part (a)

The mass will begin slipping when the combined force from the spring and the centrifugal force combine to overcome the force of static friction. The force from the spring is $F_\mathrm{s} = k\Delta L$, the centrifugal force is $F_\mathrm{c} = m\omega^2 r_i$, and the force of static friction is capped out at $F_\mathrm{f} = mg\mu_s$. We therefore find

$$F_\mathrm{s} + F_\mathrm{c} = F_\mathrm{f} \implies k\Delta L + m\omega^2 r_i = mg\mu_s \implies \omega = \sqrt{\frac{mg\mu_s - k\Delta L}{mr_i}}. \tag{5}$$

Part (b)

Unfortunately, due to the presence of kinetic friction, the energy of the mass is not conserved, so we cannot straightforwardly use the law of conservation of energy $E_f = E_i$. However, we *can* modify it to account for the energy lost to friction:

$$E_f = E_i - (\text{Energy Lost to Friction}). \tag{6}$$

The radius at which the mass is when the spring is at its rest length is $r_i + \Delta L$. It is convenient to call this radius r_0. The initial energy comes from the rotational kinetic energy and the spring potential energy:

$$E_i = \frac{1}{2}mr_i^2\omega^2 + \frac{1}{2}k(r_0 - r_i)^2. \tag{7}$$

The final energy is

$$E_f = \frac{1}{2}mr_f^2\omega^2 + \frac{1}{2}k(r_f - r_0)^2 \tag{8}$$

The energy lost to kinetic fricion is obtained using the equation Work = Force $\times$ Distance:

$$\text{Energy Lost to Friction} = mg\mu_k(r_f - r_i). \tag{9}$$

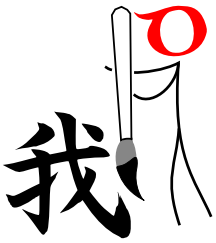
Expanding everything out, we find

$$\begin{aligned} &\frac{1}{2}mr_f^2\omega^2 + \frac{1}{2}k(r_f - r_0^2) = \frac{1}{2}mr_i^2\omega^2 + \frac{1}{2}k(r_0 - r_i^2) - mg\mu_k(r_f - r_i), \\ &\frac{1}{2}\left(m\omega^2 + k\right)r_f^2 + (kr_0 + mg\mu_k)r_f - \frac{1}{2}kr_0^2 - \frac{1}{2}m\omega^2 r_i^2 - \frac{1}{2}k(r_0 - r_i)^2 - mg\mu_k r_i = 0. \end{aligned} \tag{10}$$

We can then use the quadratic formula to find (using the larger root so that r_f is greater than r_i)

$$r_f = \frac{-(kr_0 + mg\mu_k) + \sqrt{(kr_0 + mg\mu_k)^2 + (m\omega^2 + k)(kr_0^2 + m\omega^2 r_i^2 + k(r_0 - r_i)^2 + 2mg\mu_k r_i)}}{m\omega^2 + k}. \tag{11}$$

Linguistics & Applied Sciences



5 points:

Using up to 24 of 3" by 5" lined paper index cards, and one standard 40 mm ping-pong ball, construct a "boat" that will carry as much weight as possible while floating in water. Other conditions:

- Your boat does not need to resemble an actual boat. It does need to float in water.
- No other materials can be part of your boat, including no glues or adhesives.
- You can cut or tear the cards and the ball as you want, if you want.
- As weights, you should use any coins you have access to. As part of your solution, include the denominations and quantities of coins, look up the weights on the internet and figure out the total weight to the nearest 0.1 gram.
- The weights need to rest on top of the boat and be visible.
- The weights can be placed on the boat before or after the boat is placed in water. After the final weight is placed and the boat is floating (whichever is later), you have to show in your video that your boat stays floating and upright and supports the weights for at least 30 seconds.
- Make sure that the boat is seen to not touch the bottom of your water container.
- Document as much of the construction process as you can in your video.

Hint:

No hint this month.

Solution:

Featured Solution
by Henry Kendall
https://www.youtube.com/watch?v=ch66Iptm94k

Featured Solution
by Matthew Zevakhin
https://www.youtube.com/watch?v=sSqb33atdkY

10 points:

Here are 7 sentences from a language and their corresponding English translations. **These are ALREADY MATCHED, you do not need to match them.**

Translate the following sentences into English and explain your reasoning:

1. Jae soaurbaedahlbvi?	$\longleftrightarrow$	1. *They are teachers?*
2. Sejebeirmaer ae sohrgosohrgosohrg.	$\longleftrightarrow$	2. *The girl is most likely still snoring.*
3. Le lipa pihmaer.	$\longleftrightarrow$	3. *You see the woman.*
4. Sae saeverever.	$\longleftrightarrow$	4. *We are most likely not dancing.*
5. Soaurdaedoahlmaer e pap.	$\longleftrightarrow$	5. *The worker talks.*
6. Je saebaedahl.	$\longleftrightarrow$	6. *They don't teach.*
7. Serihbvi ae saesohrg.	$\longleftrightarrow$	7. *The children most likely don't snore.*

(a) Pihbvi ae soaurdaedoahlbvi?

(b) Se saelipa soaurbaedahlmaer.

Translate the following sentences into the language and explain your reasoning:

(c) *The girls are still not dancing.*

(d) *They most likely work.*

Explain how to form the following English words in the language:

(e) *"talking"*

(f) *"dancers"*

Hint:

Here are the 7 sentences broken into morphemes (units of meaning):

1. J-ae soaur-baedahl-bvi?	$\longleftrightarrow$	1. *They are teachers?*
2. Sejebeir-maer ae sohrgo-sohrgo-sohrg.	$\longleftrightarrow$	2. *The girl is most likely still snoring.*
3. L-e lipa pih-maer.	$\longleftrightarrow$	3. *You see the woman.*
4. S-ae sae-vere-ver.	$\longleftrightarrow$	4. *We are most likely not dancing.*
5. Soaur-daedoahl-maer e pap.	$\longleftrightarrow$	5. *The worker talks.*
6. J-e sae-baedahl.	$\longleftrightarrow$	6. *They don't teach.*
7. Serih-bvi ae sae-sohrg.	$\longleftrightarrow$	7. *The children most likely don't snore.*

Solution:

Answer:
Translate the following sentences into English and explain your reasoning:
(a) Pihbvi ae soaurdaedoahlbvi? $\rightarrow$ *The women are workers?*
(b) Se saelipa soaurbaedahlmaer. $\rightarrow$ *We don't see the teacher.*
Translate the following sentences into the language and explain your reasoning:
(c) *The girls are still not dancing.* $\rightarrow$ Sejebeirbvi e saevereverever
(d) *They most likely work.* $\rightarrow$ Jae daedoahl.
Explain how to form the following English words in the language:
(e) *"talking"* $\rightarrow$ "papapap"
(f) *"dancers"* $\rightarrow$ "soaurverbvi"

Solution:

Starting with the 7 sentences:

1. Jae soaurbaedahlbvi?	⟷	1. *They are teachers?*
2. Sejebeirmaer ae sohrgosohrgosohrg.	⟷	2. *The girl is most likely still snoring.*
3. Le lipa pihmaer.	⟷	3. *You see the woman.*
4. Sae saeverever.	⟷	4. *We are most likely not dancing.*
5. Soaurdaedoahlmaer e pap.	⟷	5. *The worker talks.*
6. Je saebaedahl.	⟷	6. *They don't teach.*
7. Serihbvi ae saesohrg.	⟷	7. *The children most likely don't snore.*

Let's look at pairs 3, 4, and 6:

3. Le lipa pihmaer.	⟷	3. *You see the woman.*
4. Sae saeverever.	⟷	4. *We are most likely not dancing.*
6. Je saebaedahl.	⟷	6. *They don't teach.*

Let's guess "Le", "Sae" and "Je" are *"You"*, *"We"*, and *"They"*, respectively, since they are the shortest words in each sentence. Then "saevererer" is *"not dancing"*, and "saebaedahl" is *"not teach"*. Since both start with *"sae"* we will assume that is how verbs are negated.

Let's compare sentences 2, 4, and 7 against 3, 5, and 6 (emphasis added on "ae", "e", *"most likely"*):

1. J**ae** soaurbaedahlbvi?	⟷	1. *They are teachers**?***
2. Sejebeirmaer **ae** sohrgosohrgosohrg.	⟷	2. *The girl is **most likely** still snoring.*
4. S**ae** saeverever.	⟷	4. *We are **most likely** not dancing.*
7. Serihbvi **ae** saesohrg.	⟷	7. *The children **most likely** don't snore.*

3. L**e** lipa pihmaer.	⟷	3. *You see the woman.*
5. Soaurdaedoahlmaer **e** pap.	⟷	5. *The worker talks.*
6. J**e** saebaedahl.	⟷	6. *They don't teach.*

From this we conclude "ae" is used when there is uncertainty, while "e" means an absolute statement. Sentence 1 uses "ae" because it is a question (since questions have uncertainty).

We also notice "ae" and "e" become affixed to pronouns while remaining separate for nouns (sentence 5 vs 6, 2 vs 4).

Next we can try to figure out the nouns, using 1, 2, 5, and 7:

Plural

1. Jae soaurbaedahl**bvi**?	⟷	1. *They are teachers?*
7. Serih**bvi** ae saesohrg.	⟷	7. *The children most likely don't snore.*

Singular

2. Sejebeir**maer** ae sohrgosohrgosohrg.	⟷	2. *The girl is most likely still snoring.*
5. Soaurdaedoahl**maer** e pap.	⟷	5. *The worker talks.*

We see nouns always end in "maer" or "bvi", with "maer" being for singular and "bvi" for plural nouns.

Using this rule we can make the following matches (roots only, suffixes removed):

Language	**English**
serih	*child*
sejebeir	*girl*
pih	*woman*
soaurbaedahl	*teacher*
soaurdaedoahl	*worker*

Next we notice "baedahl" is *"teach"* and "soaurbaedahl" is *"teacher"*. "soaurdaedoahl" has a similar structure so we conclude "daedoahl" is *"work"*.

The final feature we need to understand to solve the problem is how to change the duration of actions. For that we can look at sentences 2, 4 and 7:

2. Sejebeirmaer ae **sohrgo-sohrgo-sohrg**.	$\longleftrightarrow$	2. *The girl is most likely **still snoring**.*
4. Sae sae**vere-ver**.	$\longleftrightarrow$	4. *We are most likely not **dancing**.*
7. Serihbvi ae sae**sohrg**.	$\longleftrightarrow$	7. *The children most likely don't **snore**.*

From these examples we conclude repeating X once gives "X-ing" and twice gives "still X-ing". We also see the first vowel is appended for words ending in consonants (necessary for part e).

This is enough information to get the answer as written above.

Notes:

This language is a modified version of Pingelapese, a Micronesian language with less than 3,000 remaining speakers.

Most of the words came from this paper which argues for the interpretation of "ae" and "e" used in the problem, based on interviews with native language speakers.

The primary features used in the this problem were triplication/reduplication (repeating roots of words) and evidential markers (the difference between "ae" and "e" in the solution).

Other interesting facts:

- All nouns were related to people as Pingelapese has 6 different classes of endings for different categories of nouns, which is hard to show in only 7 sentences.
- There is another type of sentence in Pingelapese called an existential sentence that is sort of a reciprocal to the question sentences and mostly unique to the language. They must use "e" as there is no uncertainty about the statement (something exists or doesn't).

Computer Science

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

---

5 points:

The Sigma Fresh rap battle is run as a round robin: every team competes against every other team once, and every such match has a winner and a loser (no ties).

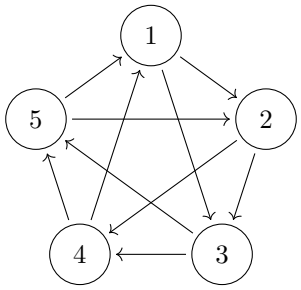
The results can be pretty confusing to interpret. However, you notice the following funny pattern: no matter what the results are, there always seems to be an ordering of the teams so that *every team in the ordering beat the team immediately before it.* That is, one can order the teams in a list $t_1, ..., t_n$ so that each team appears once in the list and, for every i from 1 to $n-1$, team t_i lost to team t_{i+1}. Such an ordering is called *Hamiltonian*, because Alexander Hamilton loved rap battles[5].

For example, suppose $n = 5$ and the ten match results are as follows, where a pair (i, j) means that team i lost to team j.

$$\Big((1,2),(1,3),(2,3),(2,4),(3,4),(3,5),(4,5),(4,1),(5,1),(5,2)\Big).$$

These results are pictured below (where an arrow $i \to j$ indicates that i lost to j). They admit several valid Hamiltonian orderings, including

$$(1,2,3,4,5) \qquad \text{and} \qquad (4,5,1,2,3) \qquad \text{and} \qquad (2,4,5,1,3).$$



(5 points) Write a program that takes a set of results and an ordering of the teams, and checks if the results are complete and the order is Hamiltonian.

[5]This detail may not be strictly true; see instead William Rowan Hamilton.

Input Your program should read the file `input.txt`. Teams are labeled $1, \ldots, n$.

- *Line 1:* Two integers n and m, where $n \geq 2$ is the number of teams and $m \geq 0$ is the number of reported match results.
- *Lines 2 to $m+1$:* Each line has two integers u v, separated by a space, with $1 \leq u, v \leq n$ and $u \neq v$, indicating that team u lost to team v.
- *Final line:* n space-separated integers t_1 t_2 $\cdots$ t_n — a proposed ordering of the team.

What to Check

- *Complete?* The results are *complete* if and only if every unordered pair $\{i, j\}$ with $1 \leq i \neq j \leq n$, one of (i, j) or (j, i) appears among the m lines, but not both.
- *Hamiltonian?* The ordering $t_1, \ldots, t_n$ is *Hamiltonian* if it includes all n teams without repetitions and, for every $i = 1, \ldots, n-1$, the pair (t_i, t_{i+1}) appears among the results (that is, t_i lost to t_{i+1}).

Output Format Write exactly two lines to the file `output.txt`

```
complete YES|NO
hamiltonian YES|NO
```

Examples

Example 1 — complete and Hamiltonian

```
5 10
1 2
1 3
2 3
2 4
3 4
3 5
4 5
4 1
5 1
5 2
1 2 3 4 5
```

Correct output

```
complete YES
hamiltonian YES
```

Example 2 — complete but not Hamiltonian

```
5 10
1 2
1 3
2 3
2 4
3 4
3 5
4 5
4 1
5 1
5 2
1 3 2 4 5
```

Correct output

```
complete YES
hamiltonian NO
```

Example 3 — incomplete but Hamiltonian ordering still holds

```
5 9
1 2
1 3
2 3
2 4
3 4
3 5
4 5
4 1
5 2
1 2 3 4 5
```

Correct output

```
complete NO
hamiltonian YES
```

Example 4 — invalid permutation for the ordering

```
4 6
1 2
1 3
1 4
2 3
3 4
4 2
1 2 2 4
```

Correct output

```
complete YES
hamiltonian NO
```

Hint:

No hint this month.

Solution:

Answer:

Solution Overview

This problem requires us to verify two properties of a graph. The solution breaks down into three modular functions that each handle a specific verification task.

Checking Completeness

A tournament is complete when every pair of teams has played exactly once. For n teams, we expect precisely $\binom{n}{2} = \frac{n(n-1)}{2}$ matches. The algorithm first verifies this, and then ensures no duplicate matches exist by tracking each unordered pair in a set. We represent each match as a pair (i, j) where $i < j$, regardless of which team won. This allows us to detect if the same two teams played twice or if any pair never played at all.

Checking if it is Hamiltonian

A proposed ordering is hamiltonian if it forms a valid path through the tournament where each team beat the previous one. We verify three conditions in sequence. First, the ordering must contain exactly n teams. Second, it must be a valid permutation, meaning each team from 1 to n appears exactly once. Third, for each consecutive pair (t_i, t_{i+1}) in the ordering, the match result (t_i, t_{i+1}) must exist in our results.

Implementation

```
"""
Tournament Results Checker

This program verifies whether round-robin tournament results are complete
and whether a proposed team ordering is Hamiltonian (each team beat the
team before it in the ordering).
"""


def read_input(filename):

    """
    Reads tournament data from the input file.

    Args:
        filename (str): Path to the input file containing tournament data.

    Returns:
        tuple: A tuple containing:
            - n (int): Number of teams
            - results (list): List of tuples (u, v) where team u lost to
   team v
            - ordering (list): Proposed ordering of teams
    """

```

```
    with open(filename, 'r') as f:
        lines = f.readlines()

    first_line = lines[0].strip().split() #parses first line to get n and m
    n = int(first_line[0])
    m = int(first_line[1])

    results = []
    for i in range(1, m + 1):
        parts = lines[i].strip().split()
        u = int(parts[0])     #parses the m following lines: each line has "u v" meaning u lost to v
        v = int(parts[1])
        results.append((u, v))

    ordering_line = lines[m + 1].strip().split()  #parses the proposed ordering from the very last line
    ordering = [int(x) for x in ordering_line]

    return n, results, ordering


def check_completeness(n, results):

    """
    Checks if tournament results are complete.

    A tournament is complete when every pair of teams has exactly one match
    result. For n teams, there should be exactly n*(n-1)/2 matches, and each
    unordered pair {i, j} should appear EXACTLY once (either as (i,j) or (j,i)).

    Args:
        n (int): Number of teams in the tournament.
        results (list): List of match results as tuples (loser, winner).

    Returns:
        bool: True if results are complete, False otherwise.
    """

    expected_number_of_matches = n * (n - 1) // 2

    if len(results) != expected_number_of_matches:
        return False #pretty straight forward

    seen_pairs = set()

    for u, v in results:
        pair = (min(u, v), max(u, v)) #creates an unordered pair representation (smaller first)

```

```
72          if pair in seen_pairs:
73              return False #checks if this pair has already been seen (
    duplicate match)
74
75          seen_pairs.add(pair)
76
77      expected_pairs = set()
78      for i in range(1, n + 1):  #verifies if we've seen all possible pairs
    by generating all possible pairs and comparing to seen_pairs
79          for j in range(i + 1, n + 1):
80              expected_pairs.add((i, j))
81      return seen_pairs == expected_pairs
82
83
84  def check_hamiltonian(n, results, ordering):
85
86      """
87      Checks if the proposed ordering is Hamiltonian.
88
89      An ordering is Hamiltonian if:
90      1. It contains all n teams exactly once (valid permutation)
91      2. For each consecutive pair (t_i, t_{i+1}) in the ordering,
92         team t_i lost to team t_{i+1} (the result (t_i, t_{i+1}) exists)
93
94      Args:
95          n (int): Number of teams in the tournament.
96          results (list): List of match results as tuples (loser, winner).
97          ordering (list): Proposed ordering of teams.
98
99      Returns:
100         bool: True if ordering is Hamiltonian, False otherwise.
101     """
102
103     if len(ordering) != n:
104         return False #pretty straight forward
105
106     if set(ordering) != set(range(1, n + 1)):
107         return False #checks if the ordering is a valid permutation (
    everyone appears once)
108
109     results_set = set(results)
110     for i in range(len(ordering) - 1): #checks each consecutive pair in
    the ordering: we know that in order for the ordering
111         current_team = ordering[i]     #to be hamiltonian, team at
    position i must have lost must
112         next_team = ordering[i + 1]
113         if (current_team, next_team) not in results_set:
114             return False
115
116     return True
117
118
119 def write_output(filename, is_complete, is_hamiltonian):
120
```

```
    """
    Writes the results to the output file.

    Args:
        filename (str): Path to the output file.
        is_complete (bool): Whether the tournament results are complete.
        is_hamiltonian (bool): Whether the ordering is Hamiltonian.
    """

    with open(filename, 'w') as f:
        f.write(f"complete {'YES' if is_complete else 'NO'}\n")
        f.write(f"hamiltonian {'YES' if is_hamiltonian else 'NO'}\n")

#and finally:

n, results, ordering = read_input('input.txt') #reading input.txt
is_complete = check_completeness(n, results) #checking completeness
is_hamiltonian = check_hamiltonian(n, results, ordering) #checking
    hamiltonianess
write_output('output.txt', is_complete, is_hamiltonian) #writing output.
    txt
```

10 points:

After reflecting on it, you realize this is not a fluke: a Hamiltonian ordering will exist for any set of tournament results! One way to prove this is by via an algorithm. Specifically, write a program that, given a set of match results, *always* finds a Hamiltonian ordering of the teams.

Input Your program should read the file `input.txt`. It has exactly the same format as in the 5pt, except that there is no final line describing an ordering. **You may assume these results are complete.**

Output Your program should write a space-separated Hamiltonian ordering of the teams to the file `output.txt`. (There may be several correct Hamiltonian orderings, you may output any of them.)

Example:

```
5 10
1 2
1 3
2 3
2 4
3 4
3 5
4 5
4 1
5 1
5 2
```

Correct output

```
1 2 3 4 5
```

Hint:

Start with a list of just two teams, and try inserting one team at a time into your list while ensuring that the ordering conditions are maintained.

Solution:

Answer: