SigmaCamp's *Problem of the Month* Contest

NOVEMBER 2025

Change for November 2025

Video submissions may now be at most **2.5 minutes long** for 5pt/10pt problems, and at most **6 minutes long** for projects.

The **POM Office Hour** is on **Saturday, December 6th** at **12PM EST**. Come ask us about the 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 **2.5 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`) submissions are accepted.

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

Project (30 points)

Linguistics

Introduction

In this project, you will be constructing your own spoken language, or as nerds on the Internet call it, *conlanging*. Points will be given for a combination of factors including your creativity, linguistic consistency, and explanation quality.

NOTE: Part (a) should not be explained in the video but only in a written component. For parts (b) and (c) all sentences in your language must be in the written component AND read aloud in the video.

Note that "spoken language" specifically excludes sign languages and other non-spoken varieties of language.

(a) **Language info sheet (WRITTEN ONLY):**
Create a language info sheet that explains the sounds, lexicon, syntax and history of your language.

i) Sounds: All languages start with sounds. Use the IPA (if you're unfamiliar, use the IPA Chart) to describe your consonants and vowels.

ii) Lexicon: Write some words in your language and their English translations. Think of what words are important in your language. For ideas of important words, see this list. Remember to include the words that you use in other parts of the project. Does your language have any interesting word structures? What about conjugations or inflections?

iii) Syntax: Explain the structure of sentences in your language. What are the rules that govern how words are combined together?

iv) History: Tell us about the history of your language. Explain how your language's origin, evolution, and developmental context have influenced its current sounds, syntax and lexicon.

(b) **Linguistics Problem Creation (WRITTEN + VIDEO):**
Create your own linguistics problem including 5–8 phrases in your language and a scrambled list of their English translations. The goal of the problem should be to match these phrases to their translations.

Make sure there is some way for your problem to be solved without prior knowledge of your language. You can include clues such as root words, plurals, parts of the word that indicate part of speech, etc.

Then, write and record a solution for your problem including the logical steps you'd expect someone to take in order to match each phrase to its translation.

(c) **Poetry (WRITTEN + VIDEO):**
Write a short (3–6 lines) piece of poetry written in a characteristic style of your language. Poetry varies across cultures and languages—think about limericks, haikus and nursery rhymes.

In your video, read your poetry and then translate it into English with an explanation of how its structure works and connects to its meaning.

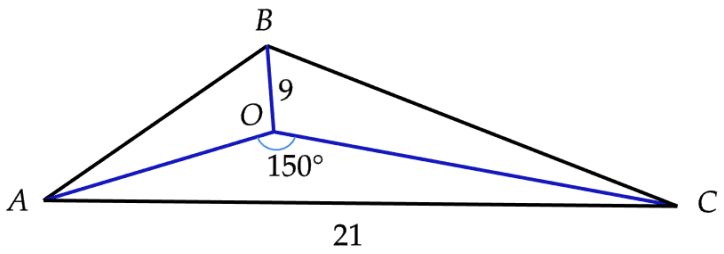
Mathematics

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.

Please show any diagrams or equations that you reference in your video, or attach them as a separate PDF.

5 points:

You are exploring a mysterious triangular island with corners labeled A, B, and C. In the *incenter* of the island is a flag, labeled O, and three straight trails lead from O to the corners A, B, and C. Each trail splits the corner it starts from into two equal angles (i.e., the trails are *angle bisectors*). Here is a picture of the island (not to scale):



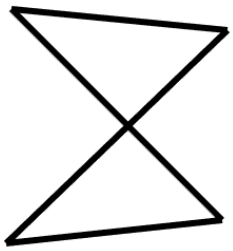
An old pirate map provides you with the following hints:

- The distance between corners A and C is 21 miles,
- The distance from the flag O to point B is 9 miles, and
- The angle at O between the trails OA and OC is $\angle AOC = 150°$.

Using these clues, find the perimeter of the island.

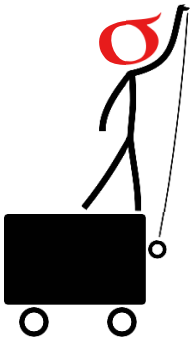
10 points:

The picture below shows a self-intersecting quadrilateral, where one of its sides intersects another side. Note that the other two sides do not intersect.



Draw a self-intersecting hexagon in which **every** side intersects **exactly one** other side. Can this be done for a heptagon? Octagon? Nonagon? Decagon?

Physics



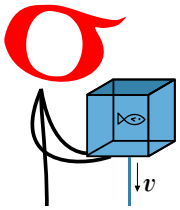
5 points:

A student is moving between apartments and is carrying a fish tank filled with water (the fish had the same density as the water and can be ignored). The fish tank develops a small leak, and water starts exiting the tank from the bottom at a speed v. Suppose the fish tank has the form of a cube with a side of $a = 20\,cm$.

(a) Determine the speed v of the water leaking out of the tank.

(b) Now imagine the student walks into an elevator with the fish tank. They notice that the leak slows down: the water now flows out *half as fast*. Determine the direction and the magnitude of the elevator's acceleration (assume the water level does not change much).

(c) How heavy would the fish tank feel to the student in the elevator compared to in the apartment?
Hint: You can find the actual mass of the tank, and compare it to its "effective mass" in the elevator.



Some Background Information. In this problem, we will explore some fundamentals of modeling fluids. A particularly useful concept when describing the motion of a fluid is *pressure*, defined as

$$P = \frac{F}{A} \tag{1}$$

where P is the pressure on some region of area A, and F is the force acting on that region.

Let's see how pressure works in a static liquid. Consider a rectangular prism filled with an incompressible liquid of density ρ, and look at the force that a block of liquid of height h exerts on the liquid below it due to gravity.

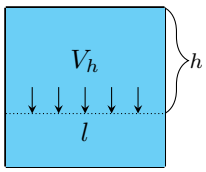


Figure 1: Side view of rectangular prism filled with a liquid of density ρ.

The volume of this block of liquid is $V = h \cdot A$, where A is the area of the bottom of the container in Figure 1. Since density is mass divided by volume (that is, $\rho = \frac{m}{V}$), we can determine that the force of gravity that pushes down on that block of water is

$$F_g = (\rho \cdot V) \cdot g = \rho \cdot h \cdot A \cdot g$$

where $g = 9.8\,\frac{\mathrm{m}}{\mathrm{s}^2}$ is the acceleration due to gravity on Earth's surface. We can use the definition of pressure in (1) to get that the pressure in a liquid is

$$P = \rho g h \tag{2}$$

where P is the pressure, ρ is the density of the liquid, h is the depth at which the pressure is measured.

Note that this pressure at a point of the liquid is exerted in all directions. The expression above then is the pressure felt by both the liquid and the walls of the container.

10 points:

For this problem, we will introduce rudimentary statistical mechanics. This is meant as an introductory problem, accessible even if you have not yet seen any statistical mechanics — the main requirement is an understanding of probability.

The central point of statistical mechanics is that we no longer think of a system in any one specific state, but as having a collection of probabilities of being in various states. For example, you may not know the precise location of a particular molecule of gas in the room you are currently sitting in, only the rough region it *might* be in. If we have many, many molecules of gas (around 10^{26} in your room), then we can average over the probabilities and find extremely precise results for the state of the entire body of gas.

A good approximation for the probability of a system to be in a given configuration was determined by Boltzmann – if the system is at a temperature T (in units of Kelvin), and a given configuration has energy E, then the probability p for the system to be in this state is proportional to

$$p \propto \exp\left(-\frac{E}{k_B \cdot T}\right). \tag{3}$$

where "$\propto$" means "proportional to", $\exp(x)$ denotes e^x, and k_B is Boltzmann's constant. The overall constant of proportionality is determined by requiring the sum of all probabilities be equal to 1.

As a simple example, suppose that a ball of mass m can be in two configurations: on a table of height h (where it has energy mgh), or on the floor (where it has energy 0). At temperature T, its probabilities are (remembering that $e^0 = 1$):

$$p_{\text{floor}} = \frac{1}{1+\exp\left(-\frac{mgh}{k_B T}\right)}, \quad p_{\text{table}} = \frac{\exp\left(-\frac{mgh}{k_B T}\right)}{1+\exp\left(-\frac{mgh}{k_B T}\right)}. \tag{4}$$

The *expected* height $\langle h \rangle$ (also the average height) is given by a weighted sum:

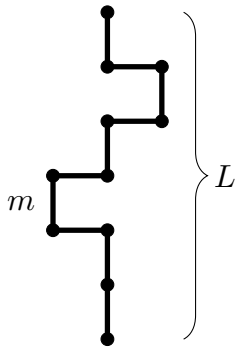
$$\langle h \rangle = 0 \cdot p_{\text{floor}} + h \cdot p_{\text{table}} = \frac{h\exp\left(-\frac{mgh}{k_B T}\right)}{1+\exp\left(-\frac{mgh}{k_B T}\right)}. \tag{5}$$

For some resources, see this short video or the first few pages of David Tong's notes (reading these is not necessary to solve the question).

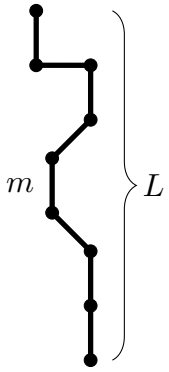
Part (a) (1 point). What happens to the average height of the ball from the example above when the temperature goes to zero? How about when the temperature goes to infinity?

See additional questions on the next page.

For the remainder of this problem, we consider a simple model for a piece of string hanging from a ceiling. We model it as a chain of N rigid rods of mass m and equal length, connected by hinges that can swing freely (see Figure 2a). For simplicity, assume that each rigid rod can only be in three configurations: horizontal pointing left, horizontal pointing right, or hanging exactly vertically.



(a) For parts (b)-(c): The diagram above shows one possible configuration of $N = 10$ rods (restricted to be either horizontal or vertical), giving the string a total length L. Note that different configurations could have different lengths (i.e. imagine a configuration having all rods vertically down).



(b) For part (d): Model of a string as N rigid rods of mass m. In part d we allow the rods to be vertical, horizontal and at 45° angles. Note that all the rods are the same length! In the diagram above, we take $N = 8$. Once again, different configurations could have different lengths.

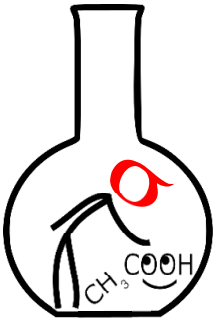
Figure 2: Simple statistical model for massive string.

Part (b) (2 points). For each rigid rod, list the three configurations it can be in and their corresponding probabilities.

Part (c) (4 points). What is the expected length of the dangling rope as a function of temperature? What is it at zero temperature? What happens to the rope as the temperature is increased?

Part (d) (3 points). Repeat (b) and (c) for the case where each rod can be in one of five configurations: horizontal left, horizontal right, at 45 degrees left, at 45 degrees right, and perfectly vertical (Figure 2b).

Chemistry

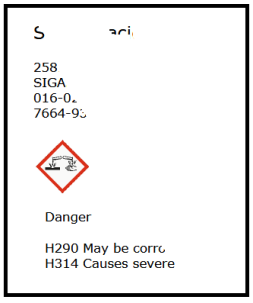


5 points:

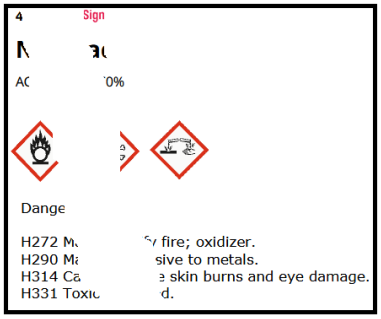
There are four acids that all fit the formula H_nXO_m, with similar molar masses (within ± 2.5 g/mol from the average). In each acid, the central atom is bonded to the same number of surrounding atoms, but as the molar mass increases, the total number of bonds increases, and so does the acidity. Identify those acids, draw their structures, and explain why these patterns occur.

10 points:

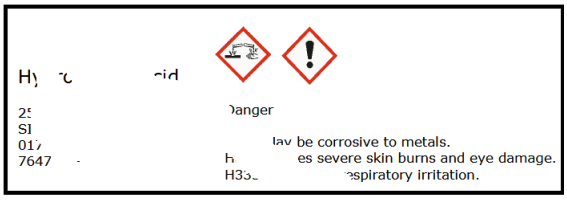
While cleaning the lab, Alice discovered three glass bottles and one polyethylene bottle at the back of a chemical storage cabinet. Unfortunately, the labels on the bottles were damaged (see Figures 1-4).



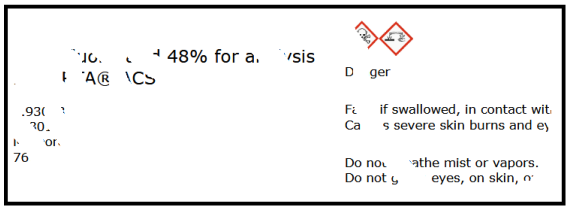
(a) Chemical 1



(b) Chemical 2



(c) Chemical 3



(d) Chemical 4

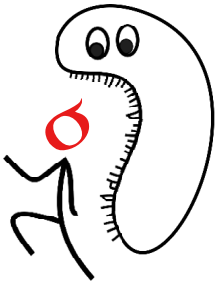
Alice wants to determine which chemical is in each bottle by reacting them with zinc metal, copper foil, and platinum wire. She determined:

- Chemical 1 reacted strongly with zinc, but did not dissolve copper until it was heated, at which point the copper began to react slowly, forming a light blue solution.
- Chemical 2 reacted vigorously with both zinc and copper, releasing dark brown fumes.
- Chemical 3 reacted vigorously with zinc, but did not react visibly with copper, even when heated.
- Chemical 4 (in the polyethylene bottle) reacted slowly with zinc and did not react visibly with copper. Alice left the glass beaker containing copper and Chemical 4 on the bench overnight to see whether a slow reaction would occur. When she returned, she found that the chemical had etched through the beaker, and all of the liquid had spilled onto the table.

As Alice expected, platinum did not react with any of the chemicals. However, when she mixed chemicals 2 and 3, platinum reacted with this mixture quite actively.

What compounds is each bottle likely to contain? What other tests could be used to confirm their identities?

Biology



5 points:

Why did many organisms develop to be aerobes when, during the emergence of life, they did not require oxygen at all? Why did some not evolve this ability?

10 points:

During the early stages of evolution, when there was no oxygen, there were still plenty of energy sources. Later, as all old sources of energy became depleted, the biosphere switched to solar light as the major source of energy. The predominant mechanism of acquiring energy became photosynthesis, accompanied by the production of large amounts of oxygen as a byproduct. In this situation, three groups of organisms resorted to three different strategies of adaptation to these new conditions. Name these groups and explain what these strategies consisted of.

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!**

When writing arithmetic expressions, we often write the symbol representing the operation in between its two arguments. For example, the sum of 1 and 2 is written as `(1 + 2)`. If we had three baskets with a pear and two apples in each basket, the total amount of fruit would be written as `(3 * (1 + 2))`. This notation, keeping the operator in between its arguments, is called *infix notation*.

It is also possible to place the operator symbol before the expression. Our two examples would be written as `(+ 1 2)` and `(* 3 (+ 1 2))`. This is called *prefix notation*. As you may have guessed, there is also *postfix notation*, which places the operator after the arguments. Our two examples would be `(1 2 +)` and `(3 (1 2 +) *)`.

5 points:

The remarkable thing about postfix and prefix notation, is that the parentheses are never necessary! There is only one way to parse a postfix or infix expression even without parentheses. Due to this nice property, postfix expressions used to be popular for calculators.

For this problem, use the following formal definitions of postfix and infix. A postfix expression is either an integer, or an expression of the form "`A B op`" where `A` and `B` are postfix expressions and `op` is one of `+`, `*`, `-`, `/`. Thus, `34 9 +` is a valid postfix expression that evaluates to 43, and `34 9 + 2 -` is a valid postfix expression that evaluates to 41. An infix expression is defined similarly, but we need parentheses for the value to be well-defined. An in-fix expression is either an integer or "`(A) op (B)`", where `A` and `B` are in-fix expressions and `op` is one of the operators mentioned above. So `(34) + (9)` evaluates to 43, and `((34) + (9)) - (2)` evaluates to 41. We need the parentheses since, for example, `10 - 3 - 2` could evaluate to either 5 or 9, depending on how it is parenthesized. You can assume that all the numbers in the input are nonnegative integers.

Write a program that translates from infix to postfix. Your program should read the input file `input.txt`, which contains a single line with an infix expression. The program should write the corresponding postfix expression on the first line of the file `output.txt` (without actually evaluating any of the operators).

Sample `input.txt`:

```
((34) + (9)) - (2)
```

Sample `output.txt`:

```
34 9 + 2 -
```

10 points:

An engineer decides to write a prefix-notation calculator—a program to parse and evaluate arithmetic expressions in prefix notation—to use for their daily engineering calculations. As they use the calculator, they quickly grow tired of copying. The calculator has no memory, so intermediate values must be written down and typed back in if used in multiple places, and common approximations such as $\sin x \approx x - \frac{x^3}{6}$ have to be punched in again and again as `(- x (/ (* x (* x x)) 6))`, where each x has to be manually substituted! The engineer decides they would like to the ability to set variables and define single parameter functions within the calculator.

You are given `input.txt` which contains expressions on each line. Write a program which evaluates expressions line by line, keeping track of defined variables and functions, and writes the value of each input line on the corresponding line of `output.txt`. All expressions, even function and variable assignments, evaluate to an integer. Expressions are either:

1. Non-negative integers such as `12345` or `42`.
2. Variables, which are strings of alphabetic characters, except the words "`if`", "`set`", and "`defun`" (these will be reserved operators). These evaluate to their most recently stored value. Examples of variable names include `var` and `dolphin`.
3. An application of an operator, which is of the form

```
(<op> <subexpr1> <subexpr2> ... <subexprN>)
```

   where the operator `<op>` is one of `+`, `-`, `*`, `/`, `<`, `=`, `if`, `set`, `defun`, or any allowable variable name, and each sub-expression `<subexpr_>` is an expression itself! Examples of operator application include

```
(+ 1 2)
(= (+ 1 2) 3)
(set dolphin (* (+ 1 2) (/ (* 6 7) 5)))
(if (= dolphin 0) (set dolphin 1) (set dolphin 0))
```

The number of sub-expressions depends on the operator. The operators are:

- The four integer operations `+`, `-`, `*`, and `/` allow 2 sub-expressions as arguments and apply the corresponding arithmetic operation. For division, return the floor of the quotient if the result is not an integer, so `(/ 3 2)` evaluates to $\lfloor \frac{3}{2} \rfloor = 1$.
- The two comparison operations `<`, `=` allow 2 sub-expressions as arguments and evaluate to 1 when true and 0 when false. For instance, `(< 1 2)` evaluates to 1, because 1 is less than 2.
- The `set` operation is always of the form `(set <var> <expr>)`. The first sub-expression, `<var>`, is always a variable name, and the second, `<expr>` is any expression. The operation evaluates to the second sub-expression `<expr>` and stores its value. Instances of the variable on subsequent lines evaluate to the stored value. For example, the expression `(set dolphin (* 6 7))` stores and returns 42. If the next line is `(+ dolphin 1)`, the sub-expression `dolphin` evaluates to 42, and the line evaluates to 43.
- The `defun` operation is always of the form `(defun <func> <param> <body>)` and always returns 0. Out of the three sub-expressions, the first two, `<func>` and `<param>` are always allowable alphabetic variable names. The expression `<body>` is any expression, which may contain instances of `<param>`. On subsequent lines, the expression `(<func> <expr>)` is an operation which takes a single sub-expression `<expr>` as an argument, and returns the value of `<body>` where each instance of `<param>` evaluates to `<expr>`. For example, the line `(defun plusfive n (+ n 5))` defines the operator `plusfive`. A subsequent line `(plusfive (* 2 2))` evaluates to 9, because the parameter `n` is set to $2 \cdot 2 = 4$, and `(+ n 5)` to evaluates $4 + 5 = 9$.
- The `if` operation is of the form `(if <cond> <ontrue> <onfalse>)`, where `<cond>`, `<ontrue>`, and `<onfalse>` are arbitary expressions. The operator evaluates `<cond>`, and returns the value of `<onfalse>`

if `<cond>` evaluates to 0, and the value of `<ontrue>` otherwise. Exactly one of `<ontrue>`, `<onfalse>` is evaluated, and variable and function definitions are only executed on the evaluated branch. For example the line `(if 1 (set dophin 42) (set dolphin 43))` executes `(set dolphin 42)`, which evaluates to 42. A subsequent line `dolphin` evaluates to 42, because the `<onfalse>` branch, `(set dolphin 43)`, was never executed.

You may assume there are no variable, function, or parameter name conflicts in `input.txt`, i.e. you may ignore the issue of variable scoping; all variables and parameters may be global. The test cases will avoid any variable conflicts, and will always use novel names for parameters.

Sample `input.txt`:

```
(+ 1 2)
(defun plustwo n (+ n 2))
(= (plustwo 1) 3)
(set dolphin (* (plustwo 1) (/ (* 6 7) 5)))
(if (= dolphin 0) (set dolphin 1) (+ 1 (set dolphin 42)))
dolphin
(defun iseven a (= a (* 2 (/ a 2))))
(defun collatzstep k (if (iseven k) (/ k 2) (+ (* 3 k) 1)))
(collatzstep dolphin)
(collatzstep (collatzstep dolphin))
```

Sample `output.txt`:

```
3
0
1
24
43
42
0
0
21
64
```