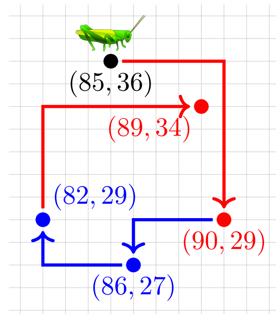
SigmaCamp's *Problem of the Month* Contest

SEPTEMBER 2023

Mathematics

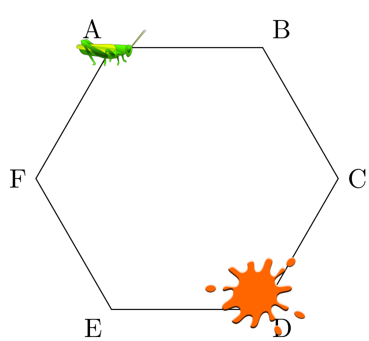
5 points:

Gerry the grasshopper lives on an infinite grid with integer coordinates. He starts at (85, 36), and his home is at (0,0). Each hop he takes, he can either go 5 in one direction and 7 in another, or 4 in one direction and 2 in another. Can he get home?



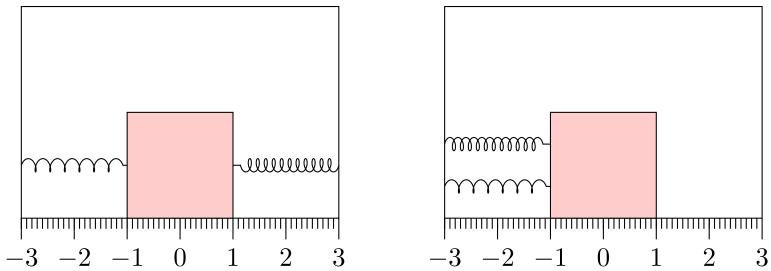
10 points:

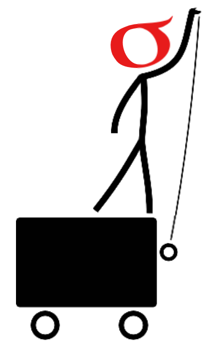
Gerry is now jumping on vertices of a regular hexagon. At each step, he jumps to one of the two neighboring vertices, choosing each possibility with probability 1/2. Gerry started at vertex A. There is a pool of orange paint at vertex D. If Gerry jumps there, he will be stained with orange paint. What is the probability that he won't be stained after 25 jumps?



Physics

5 points:





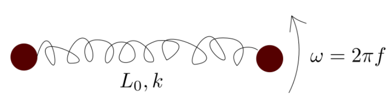
Two identical cubes of mass $m = 0.03$ kg are placed in identical frictionless boxes and attached to the boxes via Hookean springs. One cube is attached on the left and right side by springs with spring constants $k_1 = 130$ and $k_2 = 70$ N/m, respectively. The other is attached by two springs on the left side, which also have spring constants k_1 and k_2 N/m. The boxes are then rotated 90° counterclockwise. What distance are the cubes from their starting positions when they come to rest? (Assume that neither spring hits the side/bottom of the box.) You may use $g = 10$ m/s^2.

Notes:

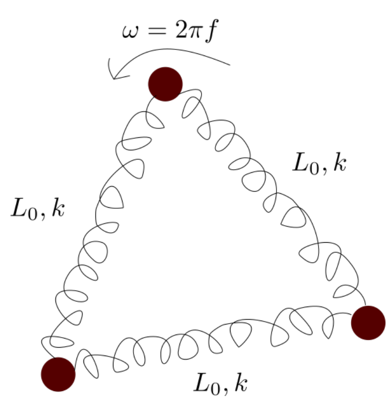
1. No points will be given for an answer without a justification.
2. Remember, if you are unfamiliar with terms like "Hookean", you are allowed to Google or look at education resources like Khan Academy. Just remember to cite your sources!

10 points:

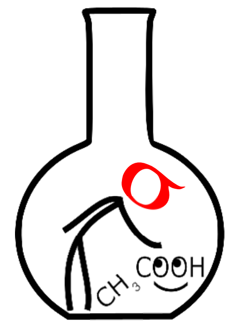
(a) Two masses of mass m are connected by a spring of rest length L_0 and spring constant k. The entire construction spins f times per second around its center of mass, with angular frequency $\omega = 2\pi f$. What is the radius of the orbit that the masses follow?



(b) This time, *three* masses of mass m are connected by springs, as shown. The springs all still have spring constant k and rest length L_0. Will the radius of orbit of the masses decrease or increase with respect to the previous case?



Chemistry



5 points:

Oh no! You didn't follow standard laboratory procedure and forgot to label four containers, each containing a different white powder. The four possible compounds are:

- Sodium carbonate
- Cupric sulfate
- Citric acid
- Lithium aluminum hydride

Since they are visually indistinguishable, you don't know which is which. Aside from these compounds, you also have access to plenty of beakers, as well as the following chemicals:

1. Hexane
2. Nitromethane
3. Benzene
4. Distilled water
5. Absolute ethanol
6. Carbon tetrachloride

Which one chemical from the list can you use to determine the identity of each of the four compounds, and how?

10 points:

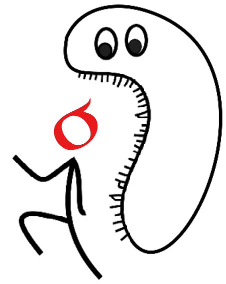
In 2023, our Sigma faculty members, Lena and Natasha, were preparing for a marine cruise. Part of the cruise research program was to measure the uptake of inorganic carbon by phytoplankton in an anoxic zone. To measure the absorption rate, the researchers needed a stock solution of labeled sodium bicarbonate ($NaH^{13}CO_3$). Lena prepared 1.1M sodium bicarbonate solution and 6.0M sodium chloride solution. According to the experimental protocol, these two solutions were supposed to be mixed directly in a syringe in a 1:1 ratio immediately before the experiment and quickly injected into a sample of ocean water taken from a depth of 500 meters. Something went wrong, however: immediately after mixing, the two solutions became cloudy, and the resulting precipitate clogged the syringe needle.

(a) Determine what compound the precipitate is.

(b) Explain why the precipitate formed.

(c) Propose a modified protocol that allows preparation of the final solution with a required concentration of the bicarbonate ion ($[H^{13}CO_3^-] = 0.55$M) and a maximal possible salinity.

Biology

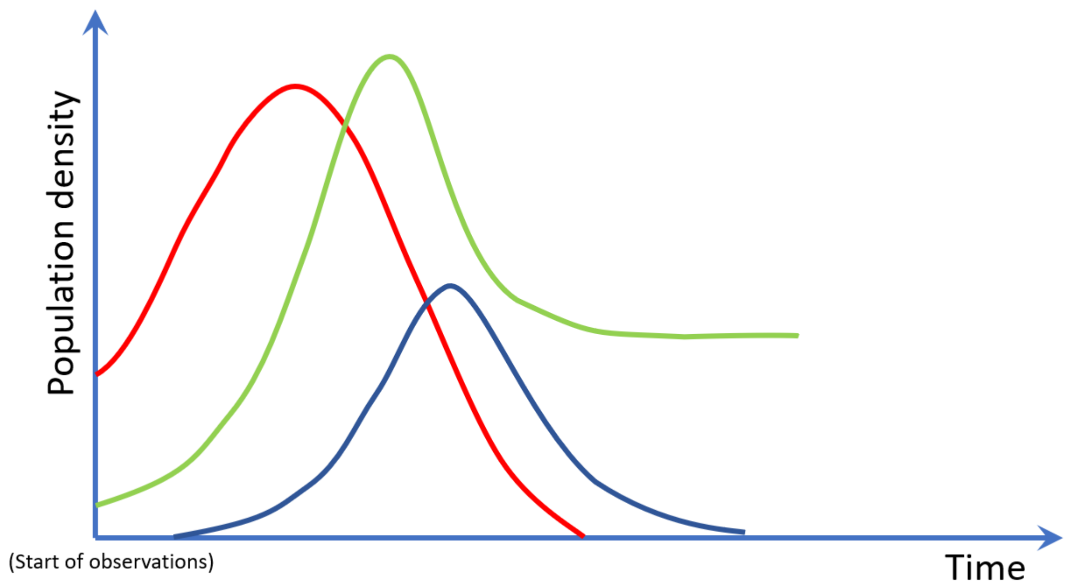


Welcome to the Biology POM 2023-2024! We're very excited to introduce you to the amazing world of Biology through some fun and thought provoking problems this year. When answering questions, we expect you to do research and consult outside sources, but you do need to cite every source you use in a consistent format (MLA 9th Edition). Here are some resources to help you with that! As always, don't hesitate to contact Sanjana Rao (sanjanarao@uchicago.edu) or Jacob Kudria (jacob@kudria.net), this year's Biology subject leads, if you have any questions.

5 points:

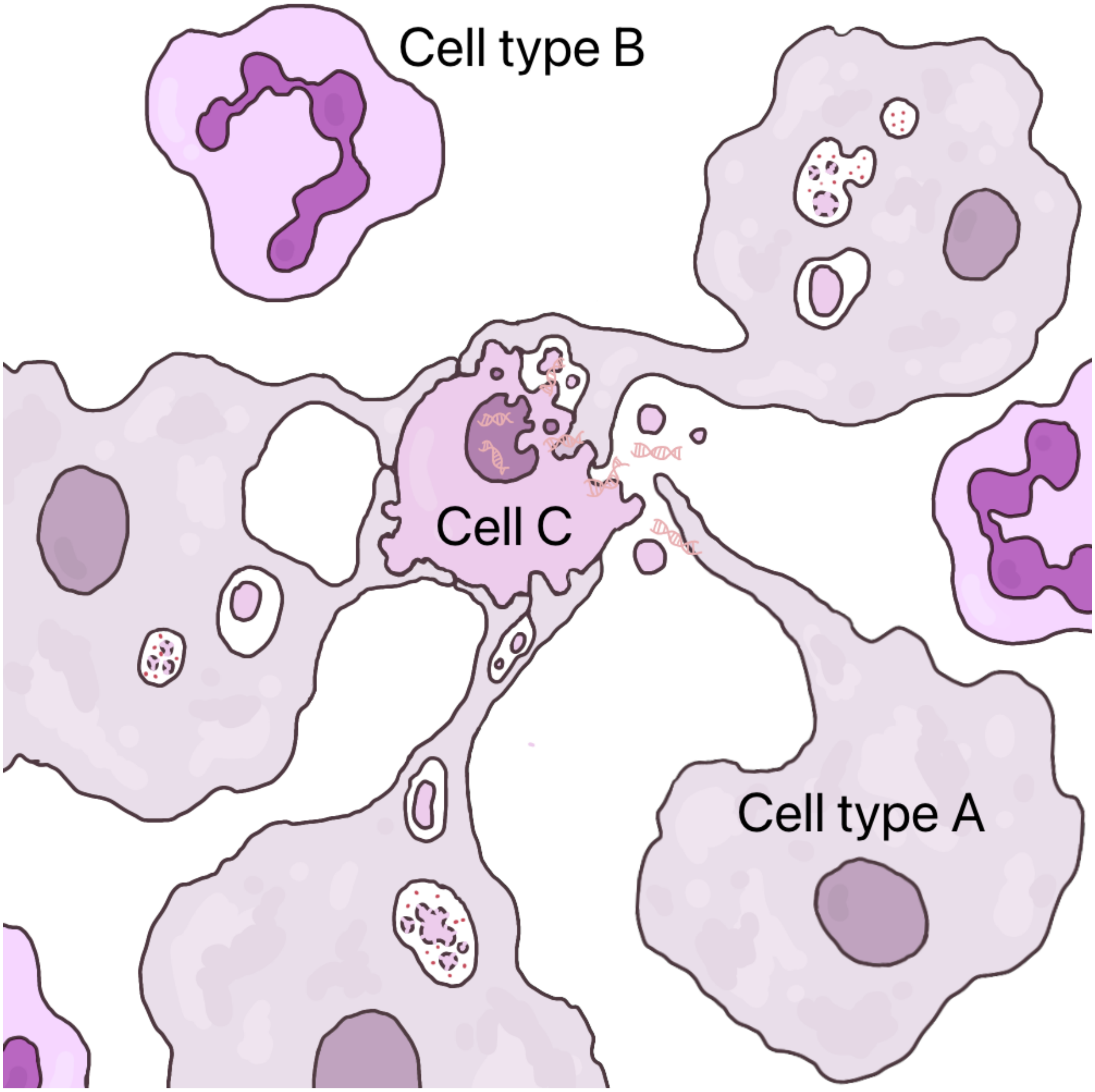
For several years, naturalists have been studying how the population of three invasive beetle species changes on a small tropical island. All three species were accidentally introduced at different times from a neighboring island. The changes in population density of the three species are summarized in the figure below.

What happened to these three species on the island and why? How did these species interact with each other and with the rest of the ecosystem? Justify your answer.



Population density of the three beetle species A (red), B (blue), and C (green).

10 points:



This depicts **a biological process involving only human cells**.

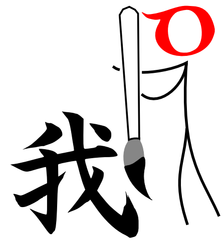
Referencing the diagram above:

(a) Identify and explain the process depicted, citing evidence from the diagram.

(b) Explain the role played by the cells of cell type A in this process, and provide two reasons why it is important.

(c) Explain how this process would be different if Cell C were a pathogen.

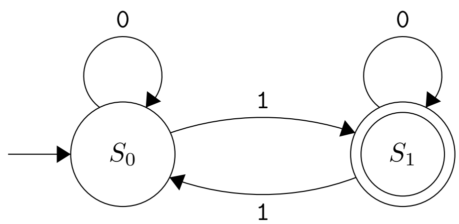
Linguistics & Applied Sciences



Have you ever wanted to try your hand at writing your own POM problem? This month, we are introducing the **Linguistics / Applied Science POM Participant Challenge**! Write and submit your own POM problem and solution by **October 10th** and it might just be published alongside our October problems.

- For inspiration, you can take a look at problems from previous months! We're looking for well written problems that are fun to solve.
- Submit your problem by emailing a PDF of the problem and solution to pom@sigmacamp.org.
- If you have any questions, you can contact pom@sigmacamp.org.

Automata theory is the study of *automata*, which are computational models used in many different areas of computer science, including the design of programming languages, artificial intelligence, and computational linguistics. This month, we will explore basic examples of automata, and how they can be used for linguistics.



An *automaton* consists of *states* and *transitions* between the states. In the above automaton, the state S_0 is the *initial state*, indicated by the arrow going into it on the left. The state S_1 is the *final state*, indicated by it being double-circled. The transitions are arrows between the states, and are labelled by `0` and `1` in this example.

A *string* is a finite sequence of characters. For the above example where the characters are `0` and `1`, an example of a string is "`011110101`". An automaton *reads* a string from left to right and follows the transitions, starting from the initial state. If the automaton finishes reading the string and ends on a final (double-circled) state, we say that the automaton *accepts* that string.

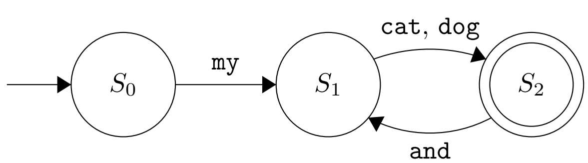
For example, the string "`01`" is accepted by the above automaton, as the automaton starts at S_0, reads `0` and loops back to S_0, then reads `1` and goes to S_1, which is a final state.

Some other strings that are accepted by this automaton (try it out for yourself!) are: `111`, `1000`, `1101101`.

Some strings that are *not* accepted are: `11`, `110101`, `11010101001`, and the empty string.

After a bit of thought, we can see that the above automaton accepts a binary string if and only if the number of `1`s in the string is *odd*.

Now let's see how we can use automata in linguistics. Instead of having our alphabet be numbers or letters, we can make them words as well!



In the above automaton, the transition with the label "`cat, dog`" allows both "`cat`" and "`dog`".

Some strings accepted by this automaton are (check it for yourself!):

- my dog
- my cat and dog
- my dog and dog and cat and dog
- my cat and cat and cat and cat and dog and cat and dog and cat and dog

Some strings that are *not* accepted by this automaton are (check it for yourself!):

- cat
- dog my
- my and cat dog
- my dog dog
- the empty string

A concise way of describing all of the strings that are accepted by this automaton is:

my <dog/cat> (and <dog/cat> ... and <dog/cat>)

where the contents inside of the parentheses ( ) are optional.

5 points:

Design and **draw** an automaton that accepts strings of the form

my <dog/cat> <eats/sleeps> (and <eats/sleeps> ... and <eats/sleeps>)

where the contents inside of the parentheses ( ) are optional.

For example, some strings your automaton should accept are:

- my dog sleeps
- my cat eats and sleeps
- my dog eats and eats and eats and eats and eats
- my cat sleeps and eats and eats and sleeps
- my cat eats and sleeps and eats and sleeps and eats and sleeps and eats and sleeps

Some strings that are should *not* be accepted are:

- cat sleeps
- my dog eats my cat
- my sleeps cat my dog
- dog cat sleeps my cat dog eats cat eats dog eats sleeps my
- the empty string

Make sure your automaton includes an initial state and a final state, and test your automaton on the above examples and come up with some other ones to ensure that it is correct.

10 points:

Design and **draw** an automaton that accepts strings of the form

```
the (<shy/cute>) <cat/dog> (<likes/hates> the (<shy/cute>) <cat/dog> who <likes/hates>
    the (<shy/cute>) <cat/dog> ...  who <likes/hates> the (<shy/cute>) <cat/dog>)
```

where the contents inside of the parentheses `( )` are optional.

For example, some strings your automaton should accept are:

- `the cat`
- `the shy dog`
- `the shy cat likes the dog`
- `the cat likes the shy dog`
- `the dog likes the cute dog who hates the lazy cat`
- `the lazy cat hates the dog who likes the cat who likes the shy cat`
- `the cat likes the dog who likes the lazy cat who hates the shy dog who likes the lazy dog`

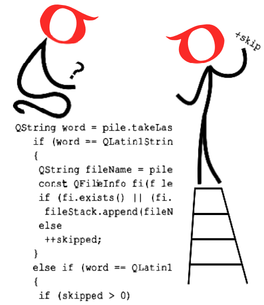
Some strings that are should *not* be accepted are:

- `the`
- `shy the dog`
- `the cute shy dog`
- `the dog hates the`
- `the dog likes the cat who`
- `the dog the shy cat`
- the empty string

Make sure your automaton includes an initial state and a final state, and test your automaton on the above examples and come up with some other ones to ensure that it is correct.

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!



We know that sometimes it can be difficult to know where to start with CS problems, so we are offering a **September Problem Solving Session!** Join us for a fun video call where we will solve a 5 point problem from a past month together, and go over some coding tips. This session will be offered on **Saturday, September 30th at 2 PM EST**, and you can join with this link: https://gatech.zoom.us/j/93510382107. Hope to see you there!

Have you ever wanted to try your hand at writing your own POM problem? This month, we are introducing the **CS POM Participant Challenge**! Write and submit your own POM problem and solution by **October 10th** and it might just be published alongside our October problems.

- For inspiration, you can take a look at problems from previous months! We're looking for well written problems that are fun to solve.
- Submit your problem by emailing a PDF of the problem statement and the Python or Java file of your solution to pom@sigmacamp.org.
- If you have any questions, you can contact pom@sigmacamp.org.
- While no points will be awarded for the Participant Challenge, depending on interest participants who have their problems published may receive a small prize.

Anna, who is working as a research assistant over the summer, is studying bacteria colony growth. For her research project, she grows bacteria colonies in a Petri dish and analyzes their characteristics. Anna wants to write a computer program to analyze the bacteria colonies. She represents a petri dish using an m-by-n grid with either a "`#`" (representing a bacterium) or "`.`" (representing empty space). Below is an example of a 10-by-4 petri dish:

```
..#.......
#.##....#.
.....#....
#.##..####
```

Anna is tasked to study the number of isolated bacteria in a petri dish, as well as the connectedness of bacteria colonies. A *bacteria colony* is a group of bacteria that are all connected to each other by a path of adjacent (horizontally, vertically, or *diagonally*) bacteria. In the below petri dish, all of the bacteria colonies are highlighted with a different color:

```
..#.......
#.##....#.
.....#....
#.##..####
```

An *isolated bacterium* is one that is not adjacent to another bacterium (i.e., it is the only bacterium in its colony). In the below petri dish, the isolated bacteria are highlighted:

```
..#.......
#.##....#.
.....#....
#.##..####
```

This month, the 5 and 10 point problems will use the same input format. Your program should receive the input file `input.txt`, which will consist of $n + 1$ lines. The first line will contain the width (m) and the length (n) of the petri dish separated by a space. The next n lines will contain m characters each, with each character either being "`#`" (representing a bacterium) or "`.`" (representing empty space).

5 points:

Write a program that receives the input as described above and computes the number of isolated bacteria within the petri dish. Your program should produce the output file `output.txt`, containing one line with the integer number of isolated bacterium.

10 points:

Write a program that receives the input as described above and computes the optimal location where a single bacterium should be added to the dish to maximize the size of the resulting largest bacteria colony.

If there is more than one position (x, y) where the bacterium can be placed, your program should compute the "least" such position: if two different positions (x_1, y_1) and (x_2, y_2) are both optimal and $x_1 < x_2$, compute (x_1, y_1). If $x_1 = x_2$ and $y_1 < y_2$, compute (x_1, y_1). Otherwise, compute (x_2, y_2). The top-left corner of the petri dish is $(0, 0)$.

Your program should produce the output file `output.txt`, containing one line with *two space-separated integers* x and y, which are the horizontal and vertical positions where adding a bacterium results in the largest possible bacteria colony.

See the examples on the next page.

Examples:

Sample 1:

`input.txt`:

```
10 4
..#.......
#.##....#.
.....#....
#.##..####
```

(5pt)
`output.txt`:

```
3
```

There are three isolated bacterium in the above dish, at $(0, 1)$, $(5, 2)$, and $(8, 1)$.

(10pt)
`output.txt`:

```
4 1
```

While another possible solution is $(4, 2)$, the y value of $(4, 1)$ is smaller, making it the correct output.

Sample 2:

`input.txt`:

```
5 6
#.#.#
.....
.#.#.
.####
#....
#.#.#
```

(5pt)
`output.txt`:

```
5
```

There are five isolated bacterium in the above dish: $(0, 0)$, $(2, 0)$, $(4, 0)$, $(2, 5)$, and $(4, 5)$.

(10pt)
`output.txt`:

```
1 1
```

The other possible solutions are $(3, 1)$ and $(3, 4)$, but the x value of $(1, 1)$ is the smallest, making it the correct output.

When we grade your submission, we test it on test cases just like these examples, so these examples are great ways to validate your responses to make sure your code is working the way it should.