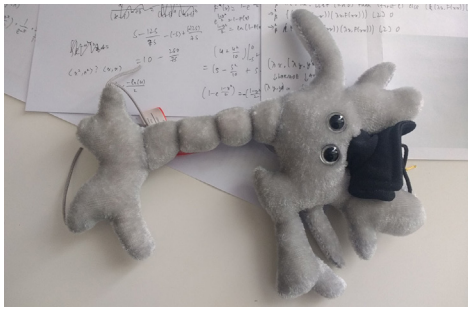
mathNEWS
VOLUME 148 • ISSUE 3
FEBRUARY 18, 2022

9 770705 041004

mastHEAD

"WHAT'S ONE THING ON YOUR BUCKET LIST?"

I had a marvellous **mastHEAD** planned out for this issue that this column is too small to contain. So instead, enjoy this picture of my sole surviving brain cell.



terrifiED
Editor, **mathNEWS**

X	Losing it all in the most expensive sample clearance lawsuit of all time
SecretSquirrel	Going to an Antarctic research station.
boldblazer	I realize that my list is not empty but I have no idea what is contained in it…
Deriving for Dick	Death.
girafarig	Finishing this article I've been working on since last like November
cy	Eating cup noodles outside while it's snowing
jeff	Dethrone Stroustrup.
CC	be a cute anime guy ✨
eternally puzzled	Why are you so interested in knowing my deepest desires?
ħ	Amend AUDB into the PMC constitution
PhilosophicalSoul	Always add more things to my bucket list
evilevievil	Save up for the future.
tendstofortytwo	Learn Japanese because anime
Skit	Getting tied up in a straitjacket and imprisoned in an abandoned mental hospital
waldo@<3.LE-GASP.ca	Free hugs and dog snuggles
cutlet	Go to sleep on time
Beyond Meta	Become a writer for the Beaverton
psychAlum	Raise a dog and cat at the same time 🐶😺
Not a N*rd	Get a degree
sqrt(cause)	Own a bucket.
sma	Go to Friesland (the land of the smas)
methNEWS	Finally come up with a pen name that doesn't make me sound like a drug addict. Then celebrate by doing copious amounts of cocaine
me	Wake up five minutes before alarm
Finchey	Wrestle a hammerhead shark in the Gulf of Mexico… wearing a Speedo.
Abald Man	Go bald
someBODY	Turn my body parts into robot parts
terrifiED	Turn up on someone else's bucket list
bipED	Scan
enamouredED	be a cute anime girl ✨
caffeinatED	Drink a bucket full of coffee.
clarifiED	I don't believe in bucket lists… but my answer's skydiving.
god⚡peED	Do comedy for a living (my pull requests don't count)

ARTICLE OF THE ISSUE

Despite the return of poorly-timed in-person midterms coinciding with prod nights (lookin' at you, MATH 136), we have no shortage of amazing articles:

- CC reports on the hypothetical Gaming Club's continued experiments with a hypothetical crypto-currency (spoiler: ~~hypo~~hyperinflation ensues)
- jeff discovers i3 in i3 while being arrested by the C++ Police for exploring tuples
- ħ breaks down the meaning of computability in a strangely existential manner
- cutlet gives the editors a headache with the character ❀
- A cabal of **mathNEWS** writers attempt to uncover the identity of the writer known as Finchey
- Lots and lots of poetry in various forms

With so many candidates to choose from, you can imagine we're having a hard time choosing an AOTI. However, after much deliberation, we have decided to award a $25 Conestoga Mall gift card to ((λ(* /)(* * /))(λ(+ -)(if - -(+ + #t)))#f) for their six-part series *An Election System Proposal*. Congratulations!

terrifiED
Editor, **mathNEWS**

Bucket lists are for losers. Pail lists are where it's at.

TERRY CHEN, mathNEWS EDITOR FOR WINTER 2022
ALONG WITH CHEN CHAI, NAMAN SOOD, CLARA XI, AND YANG ZHONG

mathASKS 148.3

FEATURING PROFESSOR LUCY GAO

QUANTUM GOOSE: WHAT IS BIOSTATISTICS ALL ABOUT?

Whenever I have to talk to border agents I tell them it's statistics but I talk to more doctors. That's actually not too far off, though! Broadly speaking, biostatisticians are statisticians who are motivated and interested in biology, medicine, and public health. Since the types of statistical methods deployed for these applications have become much more diverse, as a biostatistician you can work on just about any statistics subarea that interests you!

LOLOOLOL: WHY ARE YOU FASCINATED BY BIOSTATISTICS?

In part because of advances in biotechnology, modern biological data sets are getting increasingly large and complicated. As a result, scientists are asking the same old questions of their data but in messier settings, or they're asking more questions of their data, or they even ask brand new questions. This creates a lot of very interesting statistical challenges that are fun to tackle. Plus, I get to be involved in cool science without having to do wet lab work!

CLARIFIED: COULD YOU TELL ME MORE ABOUT THE "DOUBLE-DIPPING" PROBLEM YOU STUDY? HOW DID IT COME TO BE? AND WHAT ALTERNATIVES EXIST?

I spent my PhD thinking about clustering algorithms, which take in data corresponding to a set of objects (e.g., patients, customers, proteins) and put those objects into groups (clusters) so that the data from the objects within each cluster look similar to each other. Clustering is used in lots of applications. Examples include defining different customer subgroups to tailor marketing to, and defining subtypes of cancer.

Before you do anything with the clusters your algorithm has given you, you might want to confirm that the differences between the clusters are statistically significant. If you've taken an intro stat class then you learned about statistical tests (e.g., the t-test) you can deploy to determine if two groups have a statistically significant difference. Doing this is sometimes called "double-dipping," because you're using the data twice—once to cluster and once to test.

Unfortunately, it turns out that if you actually use the t-test on your clusters, then you'll almost always reject the null hypothesis of no difference between them, even if it's true! This is basically because you've cheated by cherry-picking your null hypothesis. You will only group objects into a cluster if their data look sufficiently different. So your data is pretty much guaranteed to look "unusual" under the null hypothesis, so far as the t-test knows.

This actually turned out to be a hard problem to solve. Simple, straightforward solutions that often work didn't. The key to the solution I proposed in my paper was to define a new p-value that acknowledges that data in different clusters will inevitably look somewhat different, when defining what's "unusual" under the null hypothesis. This spawned a whole line of follow-up work where I'm trying to tackle other similar types of "double-dipping" problems.

Anyways, point being, any of you kids out on co-ops who are doing clustering in the wild, please please please do not test for differences between the clusters! Unless you use my R package. (If you do though, make sure to tell me about it so I can put it in my tenure packet!)

SQRT(CAUSE): YOU STARTED YOUR PHD WITH AN INVESTIGATION INTO LIVER TRANSPLANTS. CAN I ASK, HOW DID IT IMPACT YOUR OPINIONS ON "FOIE GRAS"?

I— wow. Incredible question. This had literally never crossed my mind until this moment. I mean, if there's anything I learned in my time working with the gastroenterologists at Seattle Children's Hospital, it's that livers are precious and we should make the most of them. You could argue that force-feeding geese to make their livers more delicious doesn't qualify.

BOLDBLAZER: YOU WENT FROM ONE UW TO ANOTHER UW. WAS THAT PURELY A COINCIDENCE?

Absolutely not, I made it happen by conducting a job search across UWs and UWs only. (University of Waterloo, University of Washington, University of Wisconsin-Madison, University of Wyoming…)

Kidding, of course. It's just a coincidence, albeit a confusing one for me. Every time I hear "UW" my brain goes into overtime trying to figure out by context whether it's Waterloo or Washington. My only regret is that our email domain is uwaterloo.ca—can you imagine if I had gone from uw.edu to uw.ca?

ME: WHAT IS THE PROBABILITY THAT YOU AND ALICE GAO (CS INSTRUCTOR) FORM A UW UNDERGROUND RAP/SINGER DUO CALLED THE GAO GAO?

You know, there's also Jane Gao in Combinatorics and Optimization—so I'm holding out for Gao Gao Gao. Our first single could be self-titled and be set to the tune of "Bye Bye Bye" by NSYNC.

This isn't magic, this is math! Subtle but crucial difference.

PROF. DAVID MCKINNON

ON HEARTBREAK / A VALENTINE'S DAY POEM / A RESPONSE TO "ON LOVE"

Absolutely smitten has more than adequately written
A beautiful piece about being in love
When I read it I cried, and my heart deep inside
Suddenly filled with pictures of flowers and doves

But as my poems have shown, I am quite alone
And as a consequence, am unbearably bored
So my brain has spawned a poem to respond
And show single students will not be ignored

I hope that my voice can help some people rejoice
And give heart-broken readers the much needed hope
But in all honesty, this whole thing's just for me
Because we all know writing songs's how I cope

Who doesn't want to be the boy with the golden hair?
Who doesn't want to feel the love going through the air?
But sometimes unfortunately we're left with just pain to share
Leads you straight to wondering if life's always this unfair

The thrusting hand from the abyss
Will push to seethe in madness
And trust me I have been through this
Consumed myself with sadness

Ignoring all the good old things
Which used to bring you gladness
Enjoyment could turn to a fling
Pursued by ruder crassness

Then your life is in fractions, shit is just obnoxious
While they form their factions you're left to wonder options
But the one that has compassion just doesn't fit the fashion
So you take it and stash-in, ignoring all the cautions

I am of the view the universe's pretty cold
And having someone with you can finally break this mold
Or maybe lighten the load? And really brighten this world?
Could be true but take note: this is how the fantasy's sold

So time is over, clock's run out, let's snap back to reality
We're at the point in life we shape the status we call sanity
And I could spend it all imagining things that'll never be
Diverting all this energy to people that, well, aren't me

But you and I know this ain't the right way to live
So if you want to move on, you have to try to forgive
And you should really move on, unless you want to re-live
Be disturbed every day at how much thoughts can outlive

On the other side we got a path of independence
Once you leave the wrath behind you won't need repentance
You can do the math on how this will ease off your sentence
Take the socio out your path cause you're its sole attendance

And turn your efforts into shaping who you'll turn to be
Cause you're stuck with this one person for your life's eternity
Don't you want your inner person to be who outsiders see
When in their mind they see you as kind, someone happy

So this is your best chance to focus on yourself instead
Use heartbreak as your freedom, this is how you look ahead
Going solo is a power, it might not quite raise the dead
But it can raise you up until you don't need tears to shed

This song isn't meant as an attack against your character
It's advice in a canister, a metaphoric banister
Cause I'm no romance minister, I'm just here to administer
Some help for broken hearts as a Valentine's Day finisher

Even if my poem's a blight
And you're still not over it
All writers with dates on prod night
Have nothing to show for it

Now you see having a partner can cause quite the damage
So maybe knowing that you don't will prove to be a bandage
And if you're single and wonder how you will ever manage
Just don't forget that once a year you got writing advantage

Bezdomny

N FREE-TO-USE mathNEWS EDITOR NAMES THAT AREN'T PAST TENSE VERBS

- bunkbED
- birdfeED
- bloodrED
- nosebleED
- toolshED
- grapeseED
- coupdepiED
- smokeweED
- streetcrED
- evildeED
- djkhalED
- qED

writiNG

THE REAL sexNEWS 3: BROUGHT TO YOU IN HAIKU FORM, FOR SOME WEIRD REASON

This is a sure proof
Poetry will get you laid
Read and learn, virgins

❦

Q:
Dear methNEWS help me
I want to find a partner
How do I do that?

A:
If you want a mate
You must look where no one has
Deep inside your heart

Nah I'm just kidding
Fuck that shit, go join some clubs
Be more outgoing

Q:
I want to impress
My girlfriend with some sweet poems
But I cannot write

A:
Damn, she's into poems?
You should give me her number
I'll wingman for you

Q:
I wanted to go
To the sex shop down the street
But I am ashamed

A:
Never be ashamed
It's your right as an adult
No one will judge you

So go spend some cash
To help the economy
(I bought sex shop stocks)

Q:
I tried to have sex
But I did not know how to
Put on a condom

A:
Anything with sex
Is really like calculus
Practice lots at home

Q:
I am terrified
To ask out my years-long crush
Help me stop the fear

A:
You must gain some strength
And I mean that literally
Go to the gym, pal

Soon enough you will
Lift so much weight you will feel
You can conquer all

And you'll be a beast
Wait what was the prompt about?
Doesn't matter, lift!

Q:
I'm on the job grind
But my partner wants me to
spend more time with them

A:
There are many fish
In the sea but only one
Cali in the world

❦

This has been a blast
I should do this frequently
^ Things I say post-sex

methNEWS

THAT SEXUAL TENSION

Between you and the stranger when the elevator already reached your floor but the doors haven't opened yet.

Saint Valentine

Math undergrads already know too much about me.

PROF. JIM GEELEN

AN ELECTION SYSTEM PROPOSAL: PART 6

Author's note: the first part of this article was published in 147.4.

As promised in the last part of this article, I will present an acceptable victory index algorithm.

Definition 9: the *lowest losses victory index algorithm* L is a victory index algorithm defined as follows. Let $E = (S, f)$ be an election and let $s \in S$. Then,

$$(L(E))(s) = -\sum_{q \in S, f(q,s) > \frac{1}{2}} f(q, s)$$

Theorem 1: L is an acceptable victory index algorithm.

Proof: to verify condition 1, let $E = (S, f)$ be an election and let $a \in S$ be a clear winner. Therefore, for all $b \in S$, either $a = b$, in which case $f(b, a) = \frac{1}{2}$, or a beats b, which means that $f(b, a) < \frac{1}{2}$. Therefore, by definition, $(L(E))(a) = 0$. Now, let $b \in S$ such that $b \neq a$. Since a is a clear winner, $f(a, b) > \frac{1}{2}$. Therefore, by definition, $(L(E))(b) < \frac{-1}{2}$. Therefore, a is the winner of E according to L.

To verify condition 2, let $E = (S, f)$ be an election, let T be a losing set of E, and let $a \in T$. Let $W = S \setminus T$. Since T is a losing set, W is non-empty and every candidate in W beats a. Therefore, by definition, $(L(E))(a) < \frac{-|W|}{2}$. Let $w \in W$ be such that $(L(E))(w) \geq (L(E))(x) \forall x \in W$. It follows quickly from the definitions of L and of an election that

$$\sum_{x \in W} (L(E))(x) \geq \frac{-|W|(|W|-1)}{2}$$

Therefore, the mean VI of candidates in W is at least $\frac{1-|W|}{2}$. Since w is such that $(L(E))(w) \geq (L(E))(x) \forall x \in W$, it must be the case that $(L(E))(w) \geq \frac{1-|W|}{2}$. But this is greater than $\frac{-|W|}{2}$, which in turn is greater than $(L(E))(a)$. Therefore, a is not the winner of E according to L.

To verify condition 3, let $E = (S, f)$ be an election. Let T be a losing set of E. Let $Q = S \setminus T$. Let g be the restriction of f on $Q \times Q$. Let $W = (Q, g)$. Let $a \in Q$. Now, since T is a losing set, a beats every candidate in T, which means that $f(b, a) < \frac{1}{2} \forall b \in T$. Therefore, by definition, $(F(E))(a) = (F(W))(a)$.

Theorem 2: L is a good victory index algorithm.

Proof: by theorem 1, L is acceptable, so it remains only to verify the two conditions in definition 8. To verify condition 1, let $E = (S, f)$ be an election such that $a \in S$ is the winner of E according to L. Let $g : S \times S \to [0, 1]$ be a function with properties that make $Q = (S, g)$ an election. Assume $g(b, c) = f(b, c) \forall b, c \in S, a \neq b, a \neq c$ and $g(a, b) \geq f(a, b) \forall b \in S$. I need to show that a is the winner of Q according to L. Let $x = (L(E))(a)$. Since $g(a, b) \geq f(a, b) \forall b \in S$, it follows immediately from the definition of L that $(L(Q))(a) \geq x$. Now let $b \in S$ where $b \neq a$. By definition,

$$(L(Q))(b) = -\sum_{q \in S, g(q,b) > \frac{1}{2}} g(q, b)$$

There are two cases to consider: either a beats b or it does not. If a beats b, then

$$(L(Q))(b) = -g(a, b) - \sum_{q \in S, q \neq a, g(q,b) > \frac{1}{2}} g(q, b)$$

Since $g(a, b) \geq f(a, b)$ and $g(q, b) = f(q, b) \forall q \in S, q \neq a$, this is at most equal to $(L(E))(b)$. However, this must be strictly less than x, since a is the winner of E according to L. Therefore, in this case, a has a VI strictly greater than the VI of b.

In the other case, a does not beat b. Therefore, $f(a, b) \leq \frac{1}{2}$. Therefore, by definition,

$$(L(Q))(b) = -\sum_{q \in S, q \neq a, g(q,b) > \frac{1}{2}} g(q, b)$$

Since $g(q, b) = f(q, b) \forall q \in S, q \neq a$, this equals $(L(E))(b)$. However, this must be strictly less than x, since a is the winner of E according to L. Therefore, in this case, a has a VI strictly greater than the VI of b. Therefore, a is the winner of Q according to L.

To verify condition 2, let $E = (S, f)$ be an election such that $a \in S$ is not the winner of E according to L. Let $g : S \times S \to [0, 1]$ be a function with properties that make $Q = (S, g)$ an election. Assume $g(b, c) = f(b, c) \forall b, c \in S, a \neq b, a \neq c$ and $g(a, b) \leq f(a, b) \forall b \in S$. I need to show that a is not the winner of Q according to L. Let $x = (L(E))(a)$. Since $g(a, b) \leq f(a, b) \forall b \in S$, it follows immediately from the definition of L that $(L(Q))(a) \leq x$. Since a is not the winner of E according to L, there exists some $w \in S$ such that $(L(E))(w) \geq x$. There are two cases to consider: either a beats b or it does not. Using almost identical arguments as for condition 1, it can be shown that, in both cases, a has a VI less than or equal to the VI of b. Therefore, a is not the winner of Q according to L.

So there you have it. The election system can now be reduced to calculating VI for every candidate according to the lowest losses victory index algorithm, and whoever has the highest VI wins the election. I really think that using this election system will reduce or even completely eliminate the need for strategic voting. I believe that if we used this election system, the leaders elected would, on average, be a better representation of the peoples' choice.

This is (finally) the last part of this article, unless I come up with something better.

((λ(* /)(* * /))(λ(+ -)(if - -(+ + #t)))#f)

THE PRESENT

He burst through the door in a flurry of desperation and rainwater.

"Do you still have it?"

The shopkeeper was startled, but knew what he meant. She reached into the glass case and extracted a watch, dark, the dial heavy and smooth. He knew his brother would like it; he would've wanted the same for himself.

He paid the sum in cash.

Today was the last day. He hadn't known last night that he'd like to have a relationship with his brother still. But the loneliness hit him hard when he woke up, and he found that he wasn't angry anymore. Not so long after that he could barely remember what had caused their separation at all.

The beach was empty when he got there. A wall of panic closed in on him. But the ship was not due for another hour—he couldn't have left. The wet sand clung to his boots as he clambered up a rock, to gaze out at the coastline.

"What are you looking for?"

It was a girl, barely. He nearly slipped at the sound of her voice. She seemed just out of childhood, thin, with the shiftiness of a person who had known hunger.

He shook his head and turned away, but turned back a moment later.

"Did you see a small blue-and-red boat in the last hour or so?"

"Depends."

"On what?" He didn't have time for her games.

Her eyes fell to his bag. He sighed, but rummaged around and pulled out half a sandwich. She ripped into it with a relish that was difficult to watch.

"Well?"

"I need cash. I'm not going to talk for food." She spoke around a mouthful of cucumber.

The anger rose like a tidal wave. She'd already finished eating. He pulled out the remainder of his money from his purchase, but kept hold of it.

"Talk."

"At the harbour over west. They needed to refuel, or something." Her eyes didn't move from the cash as she spoke. She scampered away as soon as he handed it over, as though she expected him to change his mind.

He walked slowly now, the apprehension of seeing his brother again coming back now that the urgency was gone. The sky had darkened to the colour of a day-old bruise by the time he arrived.

He didn't have to board the boat. There was his brother on the pier, flush with the contentment of the newly married, his wife rosy and beaming beside him.

Their happiness dissolved into a careful wariness at his approach.

"I never thought it was your fault."

He needed to say it. He realized now. He'd needed to say it a long time ago, but hurt and confusion had kept him away.

His brother took a while to respond. "It seemed like you blamed me for everything that went wrong." He didn't look angry.

He nodded. He couldn't argue with that. He was young and foolish and needed a reason, and his brother had been the convenient scapegoat. His brother's words after the fact had kept him from reconciling sooner, but he didn't need the apology now.

"I have something for you."

He reached into his bag, searching for the weight of the watch. It would be his apology, his request for resolution. But it was gone. He searched once, twice. And then he realized: the girl had not made off with just the money and the food. She wouldn't be hungry tomorrow, and probably not for a long while.

He didn't know whether to cry or laugh. He would not be happy today. At least she would.

chkz

LATE NIGHT CAMPUS

Stay in SLC until midnight. The vibes are exquisite. People are quiet and the sky is a luscious black. The moon is dazzling, and you would never have known how much artificial light there is. Also, I'm alone here. Please keep me company. The all-seeing eye of St. Jerome gets closer each passing day. I fear it knows my name.

SecretSquirrel

P.S. There will be lots of janitorial noise so headphones are advised.

I. ONIONS? OH, YOU MEAN TUPLES?

Tuples are a mysterious beast in C++. My use of the word "beast" is deliberate here; it's a seriously monstrous structure, and many people recommend against using it, citing it as an example of how bloated the STL structures have become. I, for one, disagree with this sentiment, and posit that `std::tuple` is genius: a monument to creativity and innovation.

So, how does this stupid type work anyway? It works a little differently than most other STL containers, because it doesn't even count as a container in the standard. With actual containers, you can access elements with `operator[]`. For example, with `std::vector`:

```
std::vector<int> vec{1, 2, 3};
vec[0] = 99;
std::cout << vec[0] << std::endl; // prints 99
```

While construction of tuples is similar (as of C++20; before this, template parameter deduction wasn't as nice so it couldn't just implicitly deduce types from the constructor arguments), accessing tuples is a little different:

```
std::tuple tpl{3, 3.14, 'a'};
std::get<0>(tpl) = 99;
std::cout << std::get<0>(tpl) << std::endl; // prints 99
```

Well, that's an interesting way of accessing elements. It's probably indicative of something funny happening under the hood. In any case, the idea here is that we can store objects of different types in each of the slots—we just aren't allowed to change our minds about those types at run-time. I'm hiding a detail here, and it's that the type of the previous tuple isn't just `std::tuple`, it's really `std::tuple<int,float,char>`, where those template parameters are inferred by the compiler. But anyway, why can't we just use `operator[]` with tuples? What makes it so special?

Well, I mean, if you think about it, it's a hard problem, no? How would you implement it? How would you implement tuple? In particular, `operator[]` needs a return type, right? But what *would* its return type be? Let's take this thought one step further: *how would we even store the data?* With something like `std::vector` it's easy: since everything is of the same type `T`, we just store everything in an array of `T`. But what should we do here? Maybe some sort of linked list,

Oh, I can feel it now. The templates are nigh. Let's look at a possible implementation for `std::tuple`.

Now, we could do a more "traditional" linked list look, but for reasons you might see later, if you're looking to have encapsulation on your tuple class, such an approach would be nightmarish to get right with friend functions. Trust me. I tried it. It was nightmarish and I couldn't get it right. So, to make our lives easier, let's make the big idea of this *inheritance*. A sort of "linked list" of inheritance; an *onion* of inheritance, if you will. Concretely, if we had something like `Tuple<int,float,char>`, it would publicly inherit from `Tuple<float,char>`, which would publicly inherit from `Tuple<char>`, which would publicly inherit from `Tuple<>` (the "empty" tuple), each of these classes containing the data for that "node". Encapsulation is made easy by just making the data protected. In particular:

```
template<typename ... Rest>
struct Tuple { // base/empty tuple
  Tuple(Rest ... rest) {}
};

template<typename T, typename ... Rest>
struct Tuple<T, Rest ... > : public Tuple<Rest ... > {
  public:
    Tuple(T data, Rest ... rest) : Tuple<Rest ... >(rest ... ),
data{data} {}
  protected:
    T data;
};
```

This is a good start! Indeed, we have enough to create our very own tuple, and `Tuple tpl{2, 3.14, 'a'}` will produce a valid `Tuple<int,double,char>` with the data in the right places. It's still pretty useless to us though; we need a way to actually access and modify that data, i.e., we need to implement `get`.

Things brings us back to the problem we ran into before: what should the return type of `get<N>(tpl)` be? The problem was that we needed a mechanism to determine the return type for any particular `size_t N` and `Tuple tpl`. Well… structs can do everything: Let's write a struct to figure it out for us at compile-time, say `TupleTypeFinder`. Our tuples are based on the notion of an "onion" of tuple types, and to figure out the type, we'll "peel back" the inheritance-layers of that onion `N` times, and say that, whichever tuple template type we end up with, its first parameter is the type we want to return. For example, we'd have `get<1>(tpl)` make `TupleTypeFinder<int,double,char>` "peel back" the first layer of `tpl`'s type, `Tuple<int,double,char>`, to get `Tuple<double,char>`, and then we'd want the return type to be `double`, the first parameter. Let's get concrete:

```
template<size_t N, typename T, typename ... Rest>
struct TupleTypeFinder; // unspecialized declaration


template<typename T, typename ... Rest>

struct TupleTypeFinder<0,Tuple<T,Rest ... >> {
  using type = T;
}; // base case -- we've found what we wanted


template<size_t N, typename T, typename ... Rest>
struct TupleTypeFinder<N,Tuple<T,Rest>> {
  using type =
```

```
    typename TupleTypeFinder<N-1,Tuple<Rest...>>::type;
}; // recursive case -- peel back another layer
```

Basically, we're just getting this struct to peel back `N` layers to find the intended return type, and store it as a member type. With that hurdle out of the way, we're ready to write `get`, in which we'll do a similar "peeling" process to fetch the data, but use this `TupleTypeFinder` to deduce its type. Certainly, that ought to have been the biggest hurdle.

Right?

Bet.

```
template<size_t N, typename T, typename... Rest>
typename TupleTypeFinder<N,Tuple<T,Rest...>>::type&
get(Tuple<T,Rest...>& tpl) {
  return get<N-1,Tuple<Rest...>>(/* ??? */);
}

template<typename T, typename... Rest>
T& get<0>(Tuple<T,Rest...>& tpl) { return tpl.data; }
```

Yeah, an entire one of those lines is a return type. I know. Fuck off. Anyway, what should go in that `/* ??? */`? What *do* we want to pass as the argument? Well, we'd like to pass the "rest" of the tuple. How do we get the "rest" of the tuple from `tpl`?

Ah-ha. See, C++ lets us do something cute here. Remember that if our `tpl` is a `Tuple<T,Rest...>`, then it inherits from `Tuple<Rest...>`, which has its own data member. You should also remember from a certain second-year CS course that you can have a base-class reference to a derived-class, and accessing it through that reference will treat the object like one of the base-class. So, let's do just that:

```
template<size_t N, typename T, typename... Rest>
typename TupleTypeFinder<N,Tuple<T,Rest...>>::type&
get(Tuple<T,Rest...>& tpl) {
  Tuple<Rest...>& restTpl = tpl;

  return get<N-1>(restTpl);
}
```

Note that we can leave out the `Tuple<T,Rest...>` parameter since the compiler will deduce it automatically. Lovely.

Now, let's compile it… and… it doesn't compile. God. Okay, the first problem is that `get` doesn't have access to `tpl.data`; it's protected, after all. No matter, we'll thrust a little spear through our encapsulation to let it access it. Back in the `Tuple` declaration, we add `get` as a template friend function:

```
template<typename T, typename... Rest> struct Tuple<T,
Rest...> : public Tuple<Rest...> {
  /* ...the rest of the implementation... */
  template<typename R, typename... Rs>
  friend R& get<0>(Tuple<R,Rs...>& tpl;
};
```

Great, but still no dice. What's the problem, then? Huh? *Tell me!* Out with it, you god-damned lunatic, it's already been however-many pages!

The problem is subtle, but significant. As it turns out, the C++ standard does not allow partial function template specialization like we're doing for the `N == 0` case. In general. Ever. Now, back in ye olden days, I'd bring you on a side quest about something called SFINAE, and I'd show you how to implement this structure called `std::enable_if`, which only defined a member type in the case that some specified condition held true at compile time. But C++20 gives us these special things called `requires` clauses, which will do the same thing in a less verbose way. Let's toss a few of those in:

```
template<size_t N, typename T, typename... Rest
requires (N != 0)
typename TupleTypeFinder<N,Tuple<T,Rest...>>::type&
get(Tuple<T,Rest...>& tpl) {
  return get<N-1,Tuple<Rest...>>(/* ??? */);
}

template<size_t N, typename T, typename... Rest>
requires (N == 0)
T& get(Tuple<T,Rest...>& tpl) { return tpl.data; }
```

…and? It compiles? It compiles. *It compiles.* Oh, yes. A `Tuple<std::string,char,uint32_t>` for you; and yes, `Tuple<Tuple<size_t,char>,char>` for you. For everyone. What's that? You've alerted the C++ Police to my presence, you say? They're armed, you say? Ah. Hm. Interesting development.

jeff

DID EPSTEIN KILL HIMSELF?

So, I hope everyone remembers when Jeffrey Epstein was found dead. I also hope everyone remembers all the memes and other viral bits that was everywhere shortly after that news dropped. The cause of most of those memes can be attributed to the uncertainty and circumstance surrounding his death, as there was obviously going to end up being speculation over exactly what happened. One of my favourite parts of that time was when you would see a well-written paragraph that is about some topic completely unrelated to Jeffrey Epstein, but then out of nowhere, it baits you into reading a phrase that said Jeffrey Epstein didn't kill himself. However, don't let that distract you into forgetting that Thomas Bach, President of the International Olympic Committee, became the Olympic champion in fencing in 1976.

boldblazer

AWESOME POINTS II

CONTINUED FROM LAST ISSUE'S AWESOME POINTS

#ANNOUNCEMENTS

wombo combo (VP Public Relations) 7:05 PM
Hello! Gaming Club is proud to present Awesome Points (AP), a new way of rewarding club members who participate and help others out through. A big thank you to **@Arthur (VP Gaming)** for developing the AP system, and **@Josephine (VP Finance)** for getting funding for this project from ManaCorp! To get started, type `ap help` in the chat. We hope you enjoy Awesome Points! 🎉 🎉 🎉
[32 🎉] [27 😸] [5 😳]

❦

#GENERAL

Elizabeth Marcus Antonio 7:15 PM
ap help

Awesome Bot by ManaCorp [BOT ✔] 7:15 PM
Hi **@Elizabeth Marcus Antonio**! Awesome Points are a way of rewarding Gaming Club members for participating in the Gaming Club online chat. Please use the following commands to interact with the Awesome Points system.

- `ap help` brings up this help menu
- `ap tip <username>` lets you reward a fellow club member who has helped you out
- `ap status` lets you know how many Awesome Points you have
- `ap redeem` lets you redeem Awesome Points for awesome rewards!

You also gain Awesome Points for actions like chatting and reacting to messages. Keep up the awesome!
[3 😳]

Elizabeth Marcus Antonio 7:16 PM
ap status

Awesome Bot by ManaCorp [BOT ✔] 7:16 PM
Hi **@Elizabeth Marcus Antonio**! You currently have **2** Awesome Points, and can tip other members up to **10** times this hour.

Elizabeth Marcus Antonio 7:16 PM
this is sick
ap redeem

Awesome Bot by ManaCorp [BOT ✔] 7:17 PM
Hi **@Elizabeth Marcus Antonio**! You currently have **4** Awesome Points. To redeem points for one of the following prizes, use `ap redeem <prize number>`

1: $5 MathSoc CnD Gift Card (5000 points)
2: Gaming Club EXCLUSIVE T-Shirt (25000 points)
3: One Month Gaming Club GamerPro Membership (10000 Points)
4: Secret Awesome Prize (9999999 Points)

More prizes are coming soon!

Elizabeth Marcus Antonio 7:18 PM
someone tip me pls 👉 👈
i'll tip you back
[1 👍]

xXd3str0yerXx 7:22 PM
ap tip **@Elizabeth Marcus Antonio**
[1 🙏]

Awesome Bot by ManaCorp [BOT ✔] 7:22PM
@Elizabeth Marcus Antonio has been tipped 50 Awesome Points! Keep up the Awesome!

Elizabeth Marcus Antonio 7:22 PM
thanks :D
ap tip **@xXd3str0yerXx**
[1 🙏]

Awesome Bot by ManaCorp [BOT ✔] 7:22 PM
@xXd3str0yerXx has been tipped 50 Awesome Points! Keep up the Awesome!

Arthur (VP Gaming) 7:24 PM
Hey folks, glad you're enjoying the new Awesome Points system! Please try not to spam tips for no reason though, we want to reward real helpfulness and stuff like that
[3 👍] [1 🙏]

Elizabeth Marcus Antonio 7:25 PM
sorry
off topic hey does anyone want to party with me for the carnage raid tonight

it would really help me and i could use some *tips* :)
[4 👍]

❦

AP LAUNCH — T + 5 DAYS

"You *always* get the garden salad bowl, Name…" Blas groaned across the SLC table. "How do you manage to stomach that stuff? Every. Single. Day. A huge bowl of salad?"

"I like salad, and it's good for you. Do you want to try some of mine?" Name twirled a cucumber on her fork.

"No way! I'll stick to my yummy double cheeseburger, thank you very much. Everyone says I need to eat more so I can grow taller, anyways. Okay, okay, okay, this is random, but did you hear about Awesome Points?"

"I know you love Awesome Points, Blas, but we already voted not to fund the program at the last—"

"Gaming Club launched Awesome Points a couple of days ago! It's a huge hit in their online chat!"

Name put her fork down, and her brow furrowed as she pondered the new information. "How did they launch it without funding? Are the points just for fun, then?"

"Nope!" Blas' eyes gleamed with excitement. Their VP Finance, Josephine, apparently made some agreement with ManaCorp to get funding for Awesome Points."

"I… don't think they're allowed to do that without telling us. I need to look into this, Blas. Thanks for letting me know." Name's voice had turned serious and sharp.

"Awwww, Name… can't we just let Gaming Club have their fun? They've probably worked pretty hard to get this working."

"Rules are rules, Blas. And if Gaming Club is in violation of them, I'll bring them in line." Name's next bite of salad was ferocious. Blas had once heard that Name had gained the nickname 'Hawk' from somewhere, and he'd never thought it appropriate until just this moment.

❦

AP LAUNCH — T + 6 DAYS

"It's a huge success!" Josephine declared at the Gaming Club execs' weekly meeting, held around a virtual table on their Minecraft server. "I'll let Arthur tell you all about the details."

Arthur's Minecraft avatar stepped to the front of the room. "Since starting the Awesome Points program, you may have noticed our online chat has soared in popularity.

mathNEWS and Imprint have *both* run articles on us. Not just one article, but several! Allow me to read you some of the titles. 'N Ways To Earn Awesome Points Fast,' 'SQL Injection bug found in Gaming Club Chatbot,' (don't worry, we've patched that one), 'A Waste of Perfectly Good Currency: The Horrifically Flawed Concept of Awesome Points and Engagement.' If you're making enemies, then you're doing something right, right?

Over the past two weeks, one hundred and fifty six users have earned a combined total of over one hundred thousand AP, and club engagement metrics have been higher than ever! I'd like to thank everyone once again for your great work on Awesome Points, and our sponsor ManaCorp for funding the program—speaking of which, we've sent them our resume banks and they've deposited the funds in Gaming Club's ManaPay private account."

The Minecraft avatars in the room punched the air; a simulacrum of applause. Josephine continued, "thank you, Arthur! Next up, we'll have President lightSoul with updates to his long term strategic plan."

❦

"Whatcha doin', Sarah?" Wordress peered through the half-open door through which Sarah was typing frantically.

"Yaaa!"

"Chat, meet ma roommate, Wordy!! She *fantastica*. Ya, take a look, Wordy. Today, ya girl speedrunnin' Aww Some Points one-thousand-point percent! It sa new chatroom speedrun category tha just started with Gaming Club Aww Some Points." Sarah's fingers flew over the keyboard, and Wordress saw none other than Gaming Club's online chat rooms whizzing by. "Tha goal a' this run is ta earn one thousand Awesome Points as quick a' ya can! World record sa twenty mins, but I think we can do betta!"

Wordress peered over Sarah's shoulder at the curved monitor. Gaming Club's online chat was on fire. Dozens of messages poured in every second, accentuated by *ding* sounds whenever Awesome-Bot awarded points. Sarah started her own.

#L33T-GAMER-CODE

ya girl sarah two 3:39 PM
ya! can anyone help ya girl with a code problem?

xXd3str0yerXx 3:39 PM
sure thing what's up?

ya girl sarah two 3:39 PM
ya girl got two binary search tree she gotta merge….

xXd3str0yerXx proceeded to write a quick explanation which Sarah didn't even read before thanking and tipping xXd3str0yerXx some Awesome Points. xXd3str0yerXx tipped Sarah back, and Sarah let out a whoop. Wordress watched as Sarah repeated the dance with a dozen other people, and a dozen other questions, accumulating hundreds of Awesome Points.

"Now tha ya girl has five hundred Awesome Points, we can start grindin' tha Awesome Points quests for tha last half of tha run! SunlessGuy, thanks for ya donation! Keeps this girl fed, an' four months, wow!"

Wordress left Sarah's room, shaking her head and smiling as she returned to penning her short story at the kitchen table.

❦

Name didn't often frequent the Gaming Club's office, and she was surprised to see how the office had transformed. Last time, posters of the hit video game *Carnage of Glory*, which Sarah loved so much, had lined the walls, with figurines of various heroes from the game stacked on top of large, RGB-drizzled

gaming desktops. As she turned into the office today, the office was a frenzy of activity.

Gaming Club's office was very large, much to the chagrin of Name, who preferred academic clubs. It was a long, rectangular room with two rows of monitors facing each wall, and a large desk at the back, where the president of the club was seated. The back wall of the club was a floor-to-ceiling window, and Name had never quite figured out the location of the window from the outside of MC.

The president, who only went by "lightSoul" and always sent delegates to MathSoc budget meetings, was dressed in a long black cloak, bowler hat, and black mask, leaving only a sliver of pale skin and green eyes visible. Name walked straight up to lightSoul's desk, drawing glances from other club members sitting at either wall.

"Hello… lightSoul. I'm here to talk to you about Gaming Club's violation of several of MathSoc's policies. You've received an email from me regarding my concerns, and I'm here to talk to you in person about them."

lightSoul didn't speak. He swept one cloaked arm towards Josephine, who was sitting at a nearby workstation against the left wall. Josephine walked over and offered a chair.

"Hi, Councillor Person," Josephine said warmly. Name hated it when people used her last name. "Thank you for coming. As Gaming Club's VP Finance, I hope I can assuage any concerns you might have about the Awesome Points program."

Name smiled a cold smile. "I was, too."

CC

WHAT YOUR MATHSOC VPA IS DOING: PART 3

Hello! My name is Vincent, and this term I am your MathSoc Vice President, Academic.

I plan to update students in **mathNEWS** every issue about what I've been doing at MathSoc to improve the undergraduate experience in the Faculty.

So here's the highlights since last issue.

PROCTORU

I've had even more discussions with Faculty about the use of ProctorU in the online sections of STAT 230 and STAT 231 this term. If you are in either of these courses and experiences issues with your midterm, please send me an email at vpa@mathsoc.uwaterloo.ca. The more details the better.

DEAN MEETING 2

The MathSoc executives met with the Dean and the Associate Dean, Undergraduate Studies to talk about M4 and expanding the amount of student space in the Faculty.

GOING FORWARD

I'll be reaching out to Faculty on the topics of PD and work term reports in the coming weeks.

Vincent Macri
Winter 2022 MathSoc Vice President, Academic

GET A CO-OP JOB CHALLENGE

I have a challenge for you… get yo ass a motherfuckin co-op job challenge. Pick up your phone *dial*……
…….

Hi Google, yes, are you hiring today? Oh great, I will be down there and fill an application, thank you. Amazon hi, how are you? Are you hiring today? Oh great, I will be down in a momentarily to fill an application. Thank you. Hi Wish, are you hiring? Oh you are, full-stack developer! Oh great! I will be down today to fill out an application, thank you. Apple, how are you? Yes, are you guys hiring today? Oh fantastic. Can I come down today to fill an application? Oh thank you.

Challenge. Get a fucking co-op job, do that challenge.

Deriving for Dick

I MADE FUN OF SOMEONE FOR USING I3 AND THEN BECAME AN I3 EVANGELIST

Uh-huh, yeah, you heard it right. Laugh it up. Guy Makes Fun Of Thing And Then Turns Out To Actually Like The Aforementioned Thing. Story of the ages. Go away.

For context, the laptop I'm using has a screen spanning roughly 14 inches, 1080p, and I'm coming from running Openbox as a window manager with no desktop environment.

A few days ago, I saw a friend using i3 on their machine and made some light fun of them (maybe they didn't appreciate it, who knows), because i3 is a sweaty window manager lacking basic functionality from the first generation of Windows and with an over-reliance on memorizing stupid keyboard shortcuts. Why *wouldn't* you want windows to float? Floating windows let you use the same screen space however many times you want! Tiling is so much less efficient. Yes. To prove this, I would install i3 on my own computer, use it for a day, and finally point and laugh at how horrible an experience it was. I wouldn't get anything done, surely.

How wrong I was.

After installing, things mostly worked out of the box. I moved my autostart applications into the i3 config, which had pretty obvious formatting. My friend also briefly described the shortcuts to me: for most keys `N`, it's `Mod+N` to do a "normal" thing, and `Mod+Shift+N` to do a "serious" task. Sometimes they parallel each other; for example, `Mod+{J,K,L,;}` will shift focus left/down/up/right, whereas `Mod+Shift+{J,K,L,;}` will shift the window itself left/down/up/right. `Mod+N` will switch to workspace `N` for any digit `N`, whereas `Mod+Shift+N` will switch the currently-focused window to workspace `N`. This is the first window manager in which I've actually used workspaces, because it's so easy to use them and because their use is so encouraged. It just feels so much more natural to spread windows across workspaces than to always rummage through heaps of layered windows, half of which I'm not even actively using. Not using a window? Don't minimize; just throw it into a scratch workspace and take it back out if you need it later on. Want to launch an application? Don't rummage around on a desktop for it; just `Mod+D` and type what you want. The keyboard shortcuts are very minimal but powerful, and easy to remember. While I've historically dunked on people for this sort of shortcut evangelism, I've come to respect and appreciate it, because it really feels like the translation time between Wanting To Do Something and Actually Doing It is becomes almost nil with this way of doing things. It's very liberating, and it makes the overall experience much less awkward and suffocating.

Altogether, I feel is that i3 is much less *claustrophobic* than Openbox. This is partly because I'm on a laptop with one small-ish screen, and so, with Openbox, I spent a lot of my time alt-tabbing between a million windows, trying to figure out which window is which in the panel, quickly amassing many long rectangular buttons with similar appearances. With everything so small and imprecise, basic things like moving windows around and resizing them felt clumsy and clunky. I don't think Openbox supports window snapping either, so things got disorganized very quickly. I'd describe this as frustrating, I felt incapacitated, like there was a really long translation time between Wanting To Do Something and Actually Doing It. All of these are things that i3 goes the other direction in, and it works really well.

So, to my friend that I laughed at for using i3: I am sorry, and you were so right. It really is the better option.

jeff

Note: the irony is not lost on me that I'm writing this for 148.3 of **mathNEWS**.

SIGH...

There is one thing that the University of Waterloo is doing really well these days, which is being an expert in making me hate this university. Is it really the best idea to go full-steam ahead on a "return to normal" when the world is nowhere near normal? I could already tell that there would not be a normal "return to normal" ever since that decision was announced back in the Fall term. Uncertain times are not when you do stuff like that. A lot of uncertainty means you get stuff like the Omicron variant coming out of nowhere, screwing up those plans. The plan should have been to continue doing what was done for the Fall term. The university had bad enough problems as is before the pandemic even started, can one really expect things to ever go well for the university while the pandemic is still ongoing? You would think that with a background in public health, that Vivek Goel would know better than to try a "full steam ahead".

Is it possible to bring the old guy back as university president? So far, the only thing the current one is known for is basically mismanagement and errors in judgement on this entire situation. It appears to me that at least Feridun probably had a non-negative approval rating based on how people were willing to meme him (or at the very least, his name).

boldblazer

Send help I'm out of spa

A mathNEWS EDITOR WHO WOULD REALLY LIKE TO HAVE A BIGGER ROOM

II. TUPLES? OH, YOU MEAN LAMBDAS?

Psst. Hey, it's me. Listen, they know I got out; I don't have much time before they bring me back to C++ Jail. Before then, let me tell you another tale…

You remember tuples, right? Of course you do. Wait, you don't? Fine, here's a refresher:

```
#include <tuple>

auto tpl = std::make_tuple(2,std::string{"Hello"});
// this returns a std::tuple<int,std::string>

int two = std::get<0>(tpl); // access elem 0
std::string hello = std::get<1>(tpl); // access elem 1
```

I vaguely recall a discussion of how this `std::tuple` structure might be implemented in the STL, but I can't be sure… it's pretty funny, regardless. But that doesn't matter right now. I mean, really, why spend all that time talking about *tuples* when we can talk about *lambdas* instead?

Introduced in C++11, lambdas allow you to define a sort of anonymous function object, or a closure, that can be passed around like an object, and invoked like a function. It can capture variables from its surrounding scope, and it can define its own, and take parameters. For example:

```
int main() {
  int x = 3;
  int y = 9;
  auto fun = [&x,y](int z) mutable → int {
    x = 99;
    return z + x + y;
  };
  std::cout << fun(4) << std::endl; // prints 112
  std::cout << x << std::endl; // prints 99
}
```

Here, `fun` is a lambda that captures **x** by reference and **y** by copy from the surrounding scope, and takes an integer parameter **z** by value. It returns an `int`, as indicated by the `→ int` bit at the end. The `mutable` keyword indicates that it's allowed to mutate the things it captures; otherwise, they're treated as `const`. This is a pretty hefty example, but it contains most of the important things you can do with lambdas. Note that the `auto` out front is necessary, since the actual type of a lambda expression is kind of funny, for reasons which are similarly funny.

Back to tuples: *you can see it, right?* The parameter list. It's just a tuple. Type and value, both in one place. What would the *type* of our tuple be? Ah, it's not important. Let's say it's `auto`. If you need it, just use `decltype`. Let's see…

```
#include <type_traits>

template<typename ... Ts>
auto make_tuple(Ts ... vals) {
  return [vals ... ]<int N>(
    std::integral_constant<int, N>) mutable →
decltype(auto) {
      return get_impl<N>(vals ... );
    };
```

Note that the `<int N>` is a way of making the lambda *generic*, i.e., it's a template now. Furthermore, the `decltype(auto)` return type is just to make perfect forwarding work. Perfect. Now, what's happening with the parameter and return statement? Well, let's think about this. When we create a tuple with `make_tuple`, each piece of data is captured by the lambda that's returned, in order. We're going to want to retrieve that data later on using `get`, which we haven't implemented yet, but which will use `get_impl` as an in-between. The idea is that we'll be passing these arguments along perfectly to `get_impl` on evaluation, along with this **N**, which we'll have from `get`. This will be nice, because we won't have to somehow pattern match and rummage through the lambda's captures later on in another function; we can get the lambda to hand-deliver its captures to `get_impl`, and also pass along the index `N` of which element we want.

With this, the actual implementation for `get_impl` and `get` is mostly straight-forward; we'll do the usual recursive strategy and peel back layers until we have what we want, and then return it.

```
template<int N, typename T, typename ... Ts>
auto get_impl(T&& val, Ts&& ... rest) {
  if constexpr (N == 0) {
    return val;
  } else {
    return get_impl<N-1>(rest ... );
  }
}

template<int N, typename T>
auto get(T& tpl) requires requires {
  tpl(std::integral_constant<int,N>{}); } {
    return tpl(std::integral_constant<int,N>{});
}
```

Don't worry too much about the `requires requires` thing. If you want to learn more, read into concepts and constraints in C++20. In this case, it just verifies at compile-time that the expression is actually valid before letting you try to return it.

Look at this for a long time. Understand it. Seriously. What the fuck?

Good. We can use deploy our ~~monster~~ creation now.

```
auto tpl = make_tuple(2, 4.5, std::string{"jeferey"});
std::cout << get<0>(tpl) << std::endl; // 2
std::cout << get<1>(tpl) << std::endl; // 4.5
std::cout << get<2>(tpl) << std::endl; // jeferey
std::cout << get<3>(tpl) << std::endl; // COMPILE ERROR!
```

Want to pass a tuple to a function? Fine. Here's an easy way:

```
auto getFirstElem(auto& tpl) requires requires {
  tpl(std::integral_constant<int,0>{}); {
    return get<1>(tpl);
}
```

That's it. Ignore the fact that this could potentially accept some non-tuple objects. It's not airtight. It's a lambda-tuple, what are you expecting here? Also, note that the syntax I used just up there was kinda funny, eh? I used `requires requires`, but this isn't even a template function! And moreover, I just used `auto&` as a parameter to the function! Would `auto& ... tpls` have worked? Is this valid C++? Since when? Oh, I wonder… Oh no. They're coming. I need to go. I have more to say. You will see me again soon.

jeff

Disclaimer: this idea was lifted from @lefticus on Twitter. I saw this and couldn't help but share; it's just too good.

WINGS

You ever think about what it would be like to live with wings? It'd be so easy to get from place to place, you could go over top of all the buildings and it would just be great.

When you look at Minecraft, traversal gets so much easier when you gain elytra for precisely this reason. Being able to move quickly and precisely through the air is a great asset, and I feel like we as a society should therefore be working on some sort of wings or jetpacks.

Besides, doesn't everyone want to fly? Flight has been one of humankind's dreams ever since we first emerged as a species, and while we have airplanes and helicopters and such now it doesn't compare to the dream of powered flight limited to just you and your immediate vicinity, flying through the air and feeling the wind against your skin.

Sure, it'd be cold up there in the thin air, with no buildings or trees to block the wind. But bundling up would be worth it to just be able to soar to your destination.

In summary, it'd be cool to have wings. I rest my case.

Predap

MATH PROGRAMS AS WINTER OLYMPIC SPORTS

The Beijing 2022 Olympics are coming to an end. Here is a comparison of all the Math Programs with the Winter Olympic sports. No, I will not provide an explanation. If you disagree with it, you are welcome to propose a new comparison.

ActSci — Ski jumping

Applied Math — Doubles figure skating

C&O — Luge

CS — Speed skating

CS/BBA — Biathlon

Data Science — Short track speed skating

FARM — Snowboarding

First year — Ice hockey

Math/BBA — Nordic combined

Math Business — Alpine skiing

Math CPA — Freestyle skiing

Math Finance — Skeleton

Math Physics — Ice dancing

Math Studies — Men's figure skating

Math Teaching — Women's figure skating

Pure Math — Bobsled

Statistics — Cross-country skiing

Undeclared — Curling

psychAlum (Bing Dwen Dwen's super fan)

THE SUPERIOR FIRST WORDLE WORD

ADIEU

If you disagree, you're wrong. I would bid you farewell, but I instead bid you adieu.

eternally puzzled

profQUOTES

CS 246: BRAD LUSHMAN

❝ I'm literally going dumpster diving here, I've thrown it away but I still want it. And who knows what happened in the mean time!

❝ I don't mind going on side tangents, but that's an entire side vacation if you will.

❝ You're going to be destroyed anyway! You don't need your stuff anymore! I'm going to take it.

❝ Give me your data. Give me your next. It's mine now, not yours.

❝ In order to commit that theft, we need to know that our victim is about to be destroyed.

❝ Your data? Mine now.

❝ You're such high upstanding moral citizens that you don't understand the intricacies of theft.

❝ [Someone suggests killing it] No, not kill it. That's already going to happen, we don't need to get our hands dirty here.

❝ How about doing me a solid, and taking my stuff with you when you're about to be destroyed.

❝ I'm gonna take your stuff, you're gonna take my old stuff, I'm gonna live, you're gonna be destroyed, everything's good! It's actually simple.

❝ You might feel a little triggered. You might feel a little on defence. You might be offended by the compiler.

❝ While [this function] isn't wrong, it's wrong in the sense that it's morally wrong.

❝ The only answer to "why?" is because Stroustrup said so.

❝ One of the abilities of C++ is to surprise you with errors you never would've ever thought of.

PHIL 145: VANESSA CORREIA

❝ We could easily be observing a well-made robotic duck.

❝ Pi Day is in fact on March 14… hey, I got married on Pi Day! Cool.

ECE 106: SIMAR SAINI

❝ This means that no matter how we determine the value of V(r) and E(r), either by guessing, computer aided simulations, or demonic invocations, the function V(r) and E(r) is the guaranteed one we want.

CS 442: GREGOR RICHARDS

❝ You cannot convince all of Haskell that addition and subtraction are the same thing; you are only allowed to shoot yourself in the foot.

❝ As there are as many as a dozen Haskell programmers who don't have Ph.D.s[citation needed], [it] is vastly more popular than any other [pure functional] language.

❝ If you have used Haskell, great. If you haven't used Haskell, even better.

❝ I assume no mask is powerful enough to muffle my voice.

❝ I now own this course! It's mine forever now.

❝ There are more similarities than differences between these languages, because they are just Latin pronounced badly.

❝ Imperative programming in Haskell is the way a functional programming fascist would do imperative programming.

❝ I hate all programming languages.

❝ The reason you get into programming languages is not that you like them, but because you hate them and want to destroy them.

❝ I am impressed with my ability to stack overflow in Haskell, that's actually really hard to do.

CS 246: ROB HACKMAN

❝ If someone's dying, it's okay to beat them up and steal their stuff. You just have to make sure not to leave any traces… I suppose that's validly true in life as well, as long as you don't get caught.

❝ Perhaps we should beat up this dying object, steal its stuff, and toss all our trash on it. […] So we make it clean up our garbage for us.

PMATH 370: STEPHEN NEW

❝ …If the function is understood [in this notation], then you can leave the f off.

CS 341: ARNE STORJOHANN

❝ If you ever present a paper at a conference, never show the code, everyone will fall asleep right away.

SDS 131R: THERESA ROMKEY

❝ I love my job. I would do it for free. Don't tell my boss, I'm well-paid.

> After watching [Fight Club], I just wanted to beat somebody up. Didn't help with my pacifism.

> I might be a feminist, I might be a pacifist, but I will hunt you down if you steal my pen.

> If it's on your bucket list to be a fascist leader, I'm gonna teach you in this course how. *(class laughs)* No, no, it's nice to have goals!

> No… you don't want no stinking freedom!

THE PEOPLE I MET AT MY ORIENTATION

- A nice fellow who let me have a water bottle because I forgot to bring one.
- An orientation leader who later had a heart-to-heart talk about the expectations she tried to fulfill.
- My residence next-door neighbour whose local address was the only thing we shared in common.
- A student who didn't rearrange his schedule.
- My friend who threw me a birthday party the next term.
- My friend who drinks more Diet Coke than necessary.
- My friend who I had cried to after I lost my virginity in a dingy situation.
- My friend who I thought was gay and is now the straightest person I know.
- An engineer who watched me butcher "Gotta Go My Own Way" at a Karaoke mic.
- A handful of students who I played Anomia for the first time with.
- An orientation leader who followed me and my friends as we went in circles around the math buildings taking pictures of random doors.
- An orientation leader/my friend who inspired me to volunteer at the MathSoc office, apply to be an Orientation Leader, and audition for a musical.
- My friend who I sat next to for multiple classes and help me pass my MATH 135 Assignments.
- A friend that I ran into just last week after not seeing her for 2 years.
- An orientation leader who was the dealer when we played Blackjack.

Deriving for Dick

13 doesn't exist.

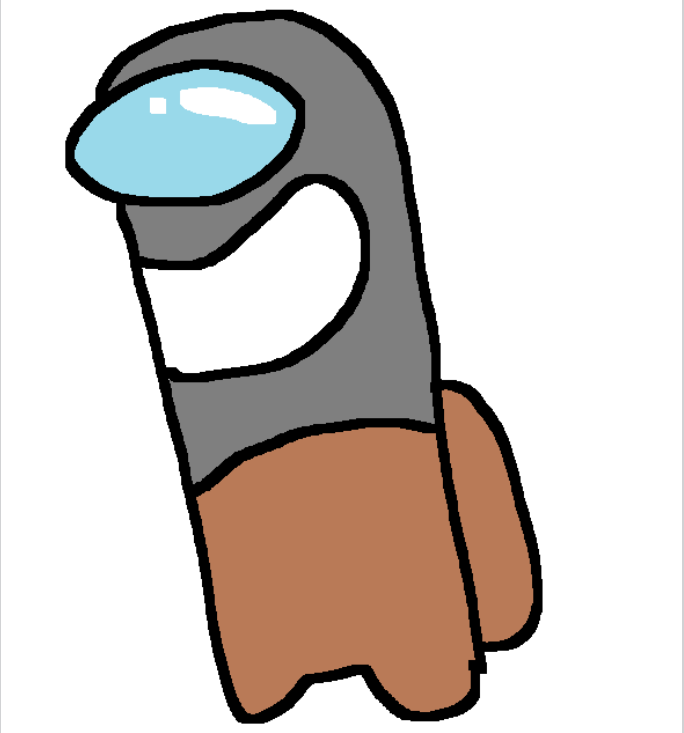
ROB HACKMAN

N THINGS YOU COULD MENTALLY BE INSTEAD OF DEAD

- the orchestral hit in a 90s boy band song
- strawberry milk
- the cause of all entropy
- lying down in a tim hortons parking lot at 3:29 am
- someone living in Dryden, Ontario, on January 19, 2006
- That Moment™ in Infinity Train S3E5
- someone with the audacity to diss English songwriter, singer, music producer and political activist Declan McKenna
- that one moment listening to "Two Trucks" by Lemon Demon and finding out what it was
- Irish stage/voice actor (and heartthrob) Shane Quigley Murphy
- screaming
- the emoji ball gags Denial is searching up while sex toy bingo happens
- unrelated anecdote but i remember when sex toy bingo was happening at st. pauls university college and i stole a balloon from the front desk and felt bad about it and returned it

Skit

A MONGOOSE



Cix

III. LAMBDAS? OH, YOU MEAN FUNCTORS?

Shut up. The C++ Police are close. I can hear them. The editors too. They're going to string me up in the plaza and kill me. But first, I have one more secret I am to bestow unto you. Listen carefully. Okay:

Remember lambdas? How do they work? Why? Who cares? Me. I mean, those must've been pretty hard to implement, eh? Pretty radical, being able to just have a function be an object like that. I bet they went to some lengths to write that into GCC.

Wrong. Wrong; wrong; wrong. You know how lambdas act like objects, right? Well, I shouldn't say they *act like objects*—rather, they *are* objects. Okay, then an object needs a type. We've been glossing over this fact by just declaring our lambdas with `auto` all the time, but that seems rather obstructionist, hm? Let's take a look.

```
int main() {
  int x;
  auto fun = [=]() mutable → void {
    ++x;
    std::cout << __func__ << ": " << x << std::endl;
  };
  fun(); fun(); fun();
}
```

Compile this code. What does it print?

```
operator(): 1
operator(): 2
operator(): 3
```

Oh, neat, the modified copy-capture of `x` gets preserved across invocations of the lambda. Also, hey, I thought that `operator()` was a member function given to classes to make them act like… oh… oh no. Computer, *enhance*.

```
int main() {
  int x = 0;
  auto fun = [=]() mutable → void {
    ++x;
    std::cout << __PRETTY_FUNCTION__ << std::endl;
  };
  fun(); fun(); fun();
}
```

Compile this code with GCC. Run it. Tell me what it prints.

```
main()::<lambda()> mutable: 1
main()::<lambda()> mutable: 2
main()::<lambda()> mutable: 3
```

Okay. So, you probably learned in a certain second-year CS course that classes in C++ can be fitted with an `operator()` method, which lets you "invoke" objects of the class as if they were a function. In an effort to sound like mathematicians, we call a class equipped with this method a *functor*. The nice thing about these so-called functors is that they *preserve state*. In particular, the data members of the class can be seen as state indicators to the function, so that information can be preserved across invocations. Here's a cute little example:

```
struct Functor {
  int x = 0;
  void operator()() {
    ++x;
    std::cout << __PRETTY_FUNCTION__ << std::endl;
  }
};

int main() {
  Functor fun;
  fun(); fun(); fun();
}
```

Here's what gets printed:

```
void Functor::operator()(): 1
void Functor::operator()(): 2
void Functor::operator()(): 3
```

Mmm. So you see how it is now, yes? This example was not so cute, nor was it arbitrarily chosen. This looks strikingly similar to what we had before. That's because, in fact, they're the same.

When you create a lambda, what actually happens behind the scenes is that the compiler generates a unique functor class, defining its `operator()` method by the body of the lambda. If `mutable` is not declared in the lambda, then the `operator()` method of the functor is declared `const`. All captures in the lambda become member variables for the generated functor class; those captured by copy are made direct members by copy, and those captured by reference get reference members. The functor generated by the compiler is called a *closure type*, and we say that the result of a lambda expression is called a *closure*.

Nothing new here after all, eh? It's all syntactic sugar. But at the end of the day, isn't that all a programming language ever is?

Now run; leave. They'll be prying the `type_traits` out of my cold, dead hands in no time.

jeff

That's him, officer! C++ Templates Guy!

A VIGILANT mathNEWS EDITOR

BREAKFAST SANDWICH MELANCHOLY SANDWICH

This article is a parody of Instant Noodles Resentment Noodles*, which will be published in* **mathNEWS** *149.2*

I want to eat egg and cheese breakfast sandwich…

I don't like to think about eating breakfast sandwich outside, like when jogging, at the egg fountain, possibly in Kitchener, or maybe in freezing rain weather, using some sort of portable toaster; instead, I want to eat my breakfast sandwich in a nice comfy diner served to me by a charismatic waiter/waitress when it's nice and early on a Saturday morning and the whole day and the whole wide world is laid out in front of me like those anime posters where the field of view is so wide that you can see the clouds curve.

Unfortunately, MC has no diners, I am under strict orders to Not Eat Inside, and it is 9-o-clock P.M. on a freezing Monday night. The only egg and cheese breakfast sandwich is the last one on the International News shelf and I am not even hungry. But I can't imagine what else I can possibly do on this Monday night and I've already spent like twenty minutes standing in SLC trying to decide what I would possibly want to eat at the International News cafeteria. So I'm gonna buy the breakfast sandwich. It looks so sad and limp just sitting in there by itself. But I pick it up anyways. At least it's warm…

Sigh… it's nine-thirty on Valentine's Day evening and I'm eating an egg and cheese breakfast sandwich out in the cold. The English muffin is soggy, and the cheese is really weird since it's turning black on one of the edges. I can't really taste the egg since the cheese is really strong and stinky. It's really not great, but at least it's warm, and it doesn't feel too cold out yet since I haven't been waiting outside for too long… it's so stinky… and then like thirty people start walking out of MC, talking loudly and interrupting my melancholy. How rude! One of them even stares at me (I'm standing on top of a rock for warmth) and I glare back. Normally the bracing cold and hot food smash together and self-annihilate to explode all of the melancholy but the cold is just not cold enough and the breakfast sandwich is not hot enough and the silence is really not silence enough and the surrealness is not disappearing…

I just wanna sit in a diner with some cutie and eat egg and cheese breakfast sandwich burp

BREAKFAST SANDWICH MERRIMENT SANDWICH

This article's a parody of Instant Noodles Resentment Noodles*, which'll be published in* **mathNEWS** *149.2!*

I want to eat egg 🥚 and cheese 🧀 breakfast sandwich!!

I like to think about eating breakfast sandwich everywhere, like at the egg 🥚 fountain, at the cheese 🧀 fountain, in Waterloo, or anywhere!! It would be so awesome to eat my breakfast sandwich in a nice cozy diner 😌 served to me by a really nice and really happy person 😁 when it's nice and early on a Saturday morning and the whole day and the whole wide world world 🌍 is laid out in front of me like those anime posters where the field of view is so wide that you can see the clouds 🌥 curve!!

MC doesn't have any diners 🍽 yet, but that's okay, since I'm Not Allowed 🚫 to Eat Inside 🥯 anyways! Nine PM on a brisk Monday night! I'm not really hungry, but I need to get my protein for the day and International News has just the perfect egg 🥚 and cheese 🧀 sandwich!! It's all by itself, so unique 🤪 and quirky ✨, like it was meant for me!! I spend like twenty minutes just fawning over 😍 it since it looks so cute there in the tray~ I buy the breakfast sandwich! It looks so happy 😀 and cute just sitting in there by itself. I pick it up, and it's piping hot! I decide to name it Eggy. 🥚

It's nine-thirty now on 💝 Valentine's Day 💝 evening and I decide to go out into the fresh, brisk, cold to eat Eggy. Eggy's English muffin is nice and soft, and the cheese is more colourful than usual! I knew Eggy 🥚 was special! I can't really taste the egg since the cheese is really strong and tasty 😋. It's great, and the warmth 😌 is so nice out in the cold 🥶. Then like thirty people start walking out of MC, talking loudly and and cheerily 🥰! How great! I catch one of them glance at me (I'm standing on top of a rock for fun) and I smile back 🙂. Normally the bracing cold 🧊 and hot food 🍜 are already nice, but the cold is just extra perfect enough and Eggy is extra unique enough and the cheerful chatter is so nice and I just feel euphoric~ 🏙✨

This just makes me even more excited for the day I sit in a cozy diner with a nice girl 😁 and eat a really tasty egg 🥚 and cheese 🧀 breakfast sandwich!! That would be fun~ 🙂

*e*n*a*mour*E*D

CC

What's my favourite menu item? I'd have to say it's the sausage biscuits for breakfast.

PROF. JEFFREY SHALLIT

"A computable function is one that is effectively computable."

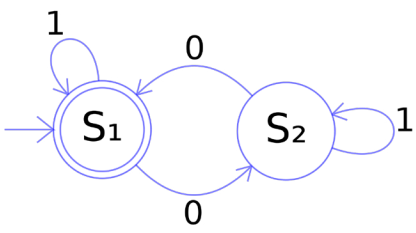
Nowadays, with so-called models of computation ranging from λ-calculi to μ-recursive functions to Turing machines (these three have been proven to be equivalent), a definition of what it means to be computable necessarily makes reference back to one of them. However, these models were developed over the course of the twentieth century, as part of the quest to formalize the notion of computability. How was the term "defined" before then?

Intuitively, "effectively computable" meant just that: it should not take any human ingenuity in order to compute. The function should be able to be carried out through a finite sequence from a finite set of instructions. The instructions must be "exact," the definition of "exact" once again appealing to intuition. Instructions, of course, should be finite. But we aren't really able to capture "effectiveness" rigorously without a model. It's tautological.

"A computable function is one a computer could compute."

One of the first attempts at a model of computation was a finite-state machine, or FSM. A machine consists of a set of states, one of which designated the initial state, a set of symbols called an alphabet, and a transition function that takes in a state and a symbol and returns a new state, if defined. If it isn't defined, we "throw an error", in whatever sense is reasonable. We designate some of the states to be final states: if the machine wants to stop it must be while it is in one of these states. Note that the set of states and the set of symbols are both finite and nonempty.

We can represent an FSM as a digraph, where vertices are states and edges valid inputs to the transition function, labeled by alphabet symbols. In the diagram stolen from Wikipedia below, notice that the unlabeled arrow into it marks `S1` as the initial state, and the double circle around it marks `S1` as a final state. We can then intuit this program as one that only succeeds in computing strings from a binary alphabet with an even number of zeroes: can you see why?



"A computable function is one that is encodable through an FSM."

Already we can probably see why this doesn't achieve our purpose satisfactorily. Consider here what the two states represent: `S1` means the string at this point has seen an even number of zeroes, and `S2` odd. Together, these are enough to handle finite binary strings of arbitrary length. However, consider now the canonical example: construct an FSM that, from an alphabet of `(` and `)`, only succeeds in computing strings where parentheses are properly balanced.

A digraph representation would see a line of states `S1` to `Sn`, `S1` initial and final. A `(` would take a state to its successor, and a `)` to its predecessor. `)` at `S1` is an undefined transition: we throw an error. And this would do the trick, given that the string had at most `n` unclosed `(`s at a time. Unfortunately, we cannot handle arbitrary strings for this problem. Finite states do not permit that.

"There are more computable functions than those that are encodable through FSMs."

Alright. That's enough dawdling. Turing machines. Same start: finite nonempty set of states, finite nonempty alphabet. Imagine now an infinite tape, each cell containing a symbol from the alphabet. We sit in a cart on one of the cells, with a given initial state. We have a state transition function: it will take in our current state, as well as the symbol in the cell we're currently occupying, and return three things: a new state, a new symbol, and a direction of left or right. We'll change our state, write the symbol onto the tape, then move. We may halt at a state if the state is final, just like an FSM. We will throw an error if the state transition function is undefined on that pair of state and symbol, just like an FSM. We will compute anything, unlike an FSM.

"A computable function is one that is encodable through a Turing machine."

The two concepts are similar, remarkably so. The critical component lies in the fact that a Turing machine may have an infinite tape. Call a Turing machine with a finite tape a finite Turing machine, which I'll abbreviate FTM. We can then ask: what is the relation between FSMs and FTMs? Suppose the FTM had tape length `k`, alphabet `A`, and set of states `S`. As any Turing machine is completely characterized by its current state, current tape, and current position on the tape, we can formulate the set of all FTMs of tape length `k` as the product of `S`, `A^k`, and `{1, ... , k}`. Let this product be the set of states of an FSM: can you see why, then, all FSMs and FTMs are in correspondence?

"A Turing-complete programming language is one that can simulate any Turing machine."

If you've coded in a language, it's probably Turing-complete. If you've written a language, it's probably Turing-complete. Near and dear to our hearts is our friend `+-<>,.[]`. We might be familiar with other fun examples: consider CS145's Gordon Villy Cormack opcode (fondly abbreviated as "gvc", "gvc-cpu", or "why is this on the final"), consisting of ten instructions and notably the ability to write itself at runtime. (Sidenote: months ago the author had planned for gvc to be the third installment of this series. They promptly abandoned the idea just minutes after launching DrRacket.)

To be pedantic, we might want to say "approximately simulate" instead. After all, physical limitations dictate that we can't make anything that's not just an FTM. We've gotten `k` pretty big though, Moore's law and whatnot, so it's good enough. We know what we mean in spirit.

Recall, now:

> *"A legitimate programming language is one that can interpret Brainfuck."*

Here, we need to be careful. While every Turing-complete language is legitimate, is the converse true? It hinges on what we mean by the word "interpret".

To be pedantic, we might want to say "approximately interpret" instead. After all, in what we have done, there is an element of preprocessing necessary to get Brainfuck to run in legitimate languages. With Desmos, we needed to change the commands into their underlying ASCII encodings. With LaTeX, the I/O wasn't exactly I/O, in the sense that the entire input "stream" was just another macro argument and the output a PDF. This is consequently not interpretation in the purest sense, but it's good enough. We know what we mean in spirit.

Take a look, for example, at vi's substitute command. `:s/PATTERN/REPLACEMENT/FLAGS` does exactly what one would expect it to do: `:s/a/b/gi` will replace every (the global flag) instance of `a` (or `A`, the case `i`nsensitive flag) in the current line with a `b`. `PATTERN` could be a regex. Now, this command by itself we could consider a full programming language, albeit a very restrictive one. To the author's knowledge, it is not Turing-complete. (Sidenote: did you know `sed` is Turing-complete?)

Nevertheless, there are possibilities. Assuming the text file vi currently has open was one with a `.c` extension, and assuming its contents were of the form `int main(){char t[10000];char*p=t;BRAINFUCK_STRING}`, what could we do? If we tried the series of substitutions of `+` to `++*p;`, `-` to `--*p;`, `>` to `++p;`, `<` to `--p;`, `[` to `while(*p){`, `]` to `}`, `,` to `putchar(*p);` and `,` to `*p=getchar();`, why — we would see that indeed,

Lemma 3.1. `:s///` is a legitimate programming language.

hu

Unicode is big — the most recent version, Unicode 14.0, has 144,697 "characters" (ranging from boring ASCII to mathematical symbols to control characters which can affect how other characters are displayed) — so it's unsurprising that it has many unusual, obscure, or storied symbols. That said, from the vast planes (no, I don't mean plains) of Unicode, I think "֎" (the "Multiocular O") in particular warrants some discussion.

֎ is a variant of the Cyrillic character O (itself a distinct Unicode character from the Latin O), and as described in a short Wikipedia article[1], appears in… at least one phrase of one 15th-century copy of the Book of Psalms of the Old Testament, with no other examples given. Furthermore, the phrase it appears in translates to "many-eyed seraphim"; or, a rough alternate translation including the character of note is "multi֎cular seraphim". Doesn't this seem like a rather low bar for inclusion in such an important and ubiquitous standard? It amounts to little more than a doodle, after all.

Well, there's a little bit more context. While there's no other evidence for the existence of ֎ given in the 2007 proposal for its addition into Unicode[2], that proposal also included other similar characters: the Monocular O (with a dot in the center), the Binocular O (with two dots in the center), and the Double Monocular O (two horizontally joined Monocular Os). Note that all of these -ocular Os are used in eye-related words. While each character individually may have had a weak case for inclusion in Unicode[3], their similar nature and appearance in more than just one text must have indicated a more concrete historical phenomenon of making eyes from Os that was, apparently, notable enough to be set in standardized stone. (More generally, there's the question of what is the "lowest bar" for inclusion in Unicode, which I'm sure is controversial in some circles I'm not a part of.)

On a more philosophical note: some Slavic religious scholar did write these characters, which I think may really be closer to doodles than symbols of strong religious significance. On some level, the writer achieved a strangely specific form of immortality. The many standards developed today are primarily constructed to make our lives more organized for (ideally, but often not in reality) the long term. I guess it's not hard to produce some long-lived information, because a lot of online content is automatically archived in various places. However, standards are not just built to last but built to remain relevant. There's a lot of future ahead of us. Who knows? A minor contribution to a project, or maybe a brief article in a student-run publication, might similarly become immortalized in its own small way.

cutlet

Author's note: thank you to the editors for ensuring the proper display of ֎!

1. https://en.wikipedia.org/wiki/Multiocular_O
2. https://www.unicode.org/L2/L2007/07003r-n3194r-cyrillic.pdf
3. As stated at http://www.unicode.org/pending/proposals.html: "The Unicode Consortium is interested in obtaining information on known glyphs, minor variants, […] however, they are generally not acceptable for character proposals."

OPEN LETTER: WHAT'S IN A NAME?

See those right-justified monikers at the end of every article? Ranging from the quotidian to the niche to the bizarre, these marks serve as the face for each **mathNEWS** writer. Such a name is not chosen lightly — and once chosen, its baggage is not easily discarded.

Why does this esteemed publication have such a tradition? Well, just think about it — do you think you'd see some of the things you see in **mathNEWS** if someone had to attach their legal name to it?

To the reader, we writers are an amorphous bunch. Long-time readers pick up on things like interests, personalities, sure. But mystique remains. Who are the *people*, the humans of flesh and bone, behind the pages? This is all complicated by the fact that, to use mathy terms, the mappings between *people* and *writer names* and *writer names* to *people* are not necessarily injective.

Perhaps on one or more occasions, you've scanned your lecture hall, letting your eyes fall on the back of someone's head, wondering, *Are they Finchey?* (Replace with the name(s) of your choice.) A fanciful little exercise if you're seeking some distraction, sure. Ultimately, it is fruitless at best and deadly at worst. "True identities," as it were, are closely guarded by this paper's editors — they are bound by hallowed obligation. Furthermore, I can only wonder what the end result of intense perseverance in this matter could possibly be. To hunt "identities" down like prey, only to gain the despicable, gluttonous gratification of solving a trite "mystery"? It's hardly something I can respect.

Beneath our chosen faces, we writers present a simulacrum of the lives of undergraduate students in this Faculty. To the average reader, I am Finchey. Nothing more and nothing less. Perhaps it's best this way.

Finchey

HUNT FOR FINCHEY

A BLURB ON mathNEWS*'S FINCHIEST WRITER*

Since December, I've been tracking down the **mathNEWS** writer known as Finchey. I don't remember how it started. By now, I don't care.

I don't know. I've downloaded every **mathNEWS** issue since January 2019, and I've read nearly every Finchey article ever written. There is no larger *why*, not anymore.

I know a lot of **mathNEWS** writers in person. I know that one of them is Finchey, and I have my suspicions. But with every new development comes a deep and unforgiving doubt. Certainty regarding facts as basic as Finchey's age and gender has come and gone. I feel contradiction. I feel a hole in my abdomen.

If I put this much energy into identifying a true stranger, I would be of great concern to the Canadian government. But Finchey is no true stranger. Finchey isn't even true: they are a collection of words. And as much as I like to pretend that art can extend beyond the page, I know deep down that Finchey is entirely limited to what their author wants to make known about themselves. So am I going too far? No: Finchey is a self-contained character, a projection of a three-dimensional body. Knowing that limits are baked into the game only makes me obsess more.

Finchey cannot be characterized by certainty. Finchey is a knowing look between people behind the veil of the page. There is a subtext, and Finchey is a projection of that subtext. Just as Freud could only understand the relationship between the conscious mind and its environment through conjecturing a third entity — the unconscious mind — I can make out the vague dance of shadows, of people living lives and forming relationships and making private references between themselves. Finchey is not the object itself, but the by-product of the object.

Why do I hunt for Finchey? I think it's funny. It's also performative, at least partially. Other writers have caught onto my search, and occasionally, I feed into the joke by reminding them that I'm still searching. In return, they feed into the joke by refusing to tell me who Finchey is (notably, Finchey's identity is an open secret). I am the object of my own joke: I am obsessing over a fictional character, and I think that that's funny. What does it say about me, that I think performative obsession is funny? What does it say about me, that I play characters around my friends?

And now I'm writing an article about it, a mostly internal **mathNEWS** matter, something of little relevance to those outside of the writership. But I guess that that's in the spirit of **mathNEWS**: articles are often written about the **mathNEWS** experience itself. It's also in the spirit of Finchey, who demonstrates great awareness of the body of work they've amassed. "Meta" is a cliché. "Self-referential" captures it more precisely. And I guess that once a project becomes large enough, self-reference just becomes reference. The events of your life inform your projects, but what happens when your life begins to coincide with the project itself? **mathNEWS** is an exercise in self-mythology. Finchey is the master of self-mythology.

I have read the text of Finchey. The text has lent itself to interpretation. Interpretation has led to theory, and theory has led to story. And now, it feels as if the stories I tell myself about Finchey overpower the text itself. I find myself overlooking details to maintain a more cohesive image of Finchey. A

tree unplugs itself from the ground and hovers through the cosmos. There are only Fincheys of the mind. I would make a terrible English major.

I took a break while writing this article. It was not a sleep: it was a seven-hour nap. I dreamt about Finchey, which is strange considering that I cannot associate a definite face or voice with them. But I dreamt that I knew who Finchey was, and I experienced a relief beyond the capacity of the conscious imagination, far exceeding the relief I'll experience when I actually learn who Finchey is. I am now awake. The hunt just got harder.

χ

UNTITLED.BMP

Is it weird to be sad about someone who makes you happy
Is it strange to want to be the only one they think about
It sounds unhealthy, right
It can't possibly be good
It's confusing
It's wrong

And I am stuck feeling that way
And I don't know what to do about these feelings
Historically they gave gone away over time
Historically I have just had to sit it out
But this feels different
But it always feels different

I try to get them out of my head but
I can't push them away from my mind
There are too many other things I don't want to think about
There is no more space for things to ignore
They are unignorable
They are unforgettable

Often it helps to get it out of my mind
Often the written word is my release
Why isn't the burden lifting then?
Why do these words make me feel more?
This should not be happening
This is against all precedent

wall outlet

ABOUT SNOW

So, I was walking back home that night, under the snow. I was breathing, sometimes inhaling, some flakes in my nose, feeling like I've been trying some new drug. I almost slipped on the ground, then I stopped walking. I watched the falling snow. It was beautiful. It was unreal. The snowflakes would gently get down from the air, arriving from everywhere, going everywhere. The streetlights would illuminate them, making them shine for a microsecond. I looked around. The ground looked like pure glitter. Everything looked brighter, and pure. The snow was covering everything. I could have walked on the lake thinking it was a field. On my arms, I could see the perfect snowflake shapes I used to cut out of paper as a kid. I closed my eyes, enjoying the feeling of snow landing on my eyelashes. I appreciated how soft the snow was this night. It was not drizzling. It was not humid, nor cold, nor heavy. It was magic from the skies. It was perfection.

Then I crossed the road. The snow, half melted, brownish on the road, splashed under my shoe. The shortcut I used to take was covered by the mountain of snow. A salt piece went under my shoe, making the rest of my walk less comfortable. The magic was gone. A snowflake landed on my cheek, melted, making me look like I was crying. It felt appropriate. I looked up again, hoping to see the beauty I knew was there. And I saw it again. They were dancing around me, and everything was dazzling again. I whispered a thank you.

PhilosophicalSoul

CALL OF DUTY DEVELOPERS MOBILIZED TO HIGH ALERT

SAN MATEO, CA—As tensions in Ukraine continue to rise, and as US intelligence sources warn that a Russian invasion is imminent, Sledgehammer Games president Aaron Halon confirmed that his staff of more than 200 game developers had been mobilized to high alert.

Halon addressed his assembled developers in the company break room, next to the foosball table. "As the possibility of conflict grows," he began, "we stand ready to make a game that isn't set in World War II for once."

According to independent investigators, Sledgehammer Games' recent actions could only be preparation for one of the largest game development efforts yet seen in modern history. Satellite imagery shows the movement of large quantities of pizza and Red Bull to Sledgehammer headquarters, which the Pentagon believes to be a primary indicator that game development is imminent.

President Biden is expected to meet with Halon next week, in an attempt to negotiate a settlement that would permanently end releases of Call of Duty games. However, most observers believe Halon is committed to inciting lacklustre single-player campaigns and repetitive multiplayer in the Western world, pointing to previous examples of Halon's reckless and belligerent behaviour, like *Call of Duty: Vanguard, Call of Duty: Modern Warfare, Call of Duty: Black Ops 4, Call of Duty: WWII,* and *Call of Duty: Infinite Warfare.*

UW Unprint

C++ LAMBDAS AREN'T CLOSURES, UNTIL THEY ARE

If you haven't already seen jeff's three-part *absolutely unhinged* series on implementing C++ tuples *from scratch* this issue, I **highly recommend** you go back and read it. If you have, in fact, managed to survive three thousand words of dense, technical C++ so psychopathic it puts the machinations of the ISO working group to shame, you may recall the following sentence from part *III. Lambdas? Oh, you mean functors?*:

> *The functor generated by the compiler is called a* closure type, *and we say that the result of a lambda expression is called a* closure.

Hmm, closures. C++ supports closures, you say? Closures like Racket and Haskell? Closures that are first-class functions, that in the (paraphrased) words of Gregor Richards, "capture the environment they are defined in, and when moved, bring their captured environment with them"?[1]

Well, yes, but also no. Welcome to the weird world of variable capturing in C++ lambdas, where everything is and is not what it seems.

C++ LAMBDAS ARE FIRST-CLASS FUNCTIONS

Yes, you read that right — in C++, you can treat lambdas exactly like you would any other value! You can return them:

```
auto is_empty_factory(const auto &container) {
  return [&]() → bool { return container.empty(); };
}
```

You can pass them in as parameters:

```
auto check_cond(const auto &cond) {
  return cond();
}
check_cond(is_empty_factory( ... ));  // works!
```

And you can store them in containers, albeit with a bit more hassle:[2]

```
#include <functional>  // define std::function
#include <vector>

// Returns a vector of boolean functions that checks if the
// corresponding parameter was empty
template <typename ... T>
auto check_conds(T&& ... containers) {
  return std::vector<std::function<bool()>>{
    is_empty_factory(containers) ... };
}

// Return vector of checks if my vectors are empty
auto check_vecs_empty() {
  auto my_vec = std::vector<int>{1, 2, 3};
  auto my_vec2 = std::vector<int>{};
  return check_conds(my_vec, my_vec2);
}
```

Notice, too, that the lambdas we've defined "capture" their environments. Indeed, the prophets were right; C++ really is a functional language.

C++ LAMBDAS, HOWEVER...

…do not automatically "bring their environment with them". And so, contrary to what jeff claims, they are not closures as we might expect.

To see what that means, consider this factory function that takes a name, and returns a lambda that says hello to that name:

```
#include <iostream>

// Say hello to name
auto say_name_factory(auto name) {
  return [&]() {
    std::cout << "Hello, " << name << "!" << std::endl;
  };
}

int main() {
  auto hello_jeff = say_name_factory("jeff");
  hello_jeff();
}
```

As with our previous examples, the lambda returned from `say_name_factory` "captures" its environment. Let's compile this program and watch it say hello back:

```
$ gcc --std=c++20 hello.cpp
$ ./a.out
Hello, 8-▪s▪▪!
```

Huh. That's not jeff's name, at least not in English. What happened here?

As it turns out, C++ lambdas do *not* capture their environments completely. The lambda we return from `say_name_factory` captures `name` by reference; however, `name` is located in the stack of `say_name_factory`. Thus, when `say_name_factory` returns and tears down its stack, it also tears down the returned lambda's reference to `name`, leaving a dangling reference. Which is why `hello_jeff` speaks in alien.

So C++ lambdas are not closures. The prophets were wrong; C++ is not a functional language.

TURNING C++ LAMBDAS INTO CLOSURES*

** with some caveats*

At this point, some programmers would admit defeat and move on with their lives. But we're C++ programmers; we can't let the realities of working with a stack-based language stop us now. Bjarne Stroustrup gave us extraordinary power when he

invented C++, and by God are we going to take this power and snort it until even the most ardent of Haskell devotees agree that C++ is a functional language.

One obvious way to get a lambda to completely capture its environment is to have it capture by copy. This means that any variables we reference inside the lambda body are copied into the lambda's environment, thus preserving them:

```
// Say hello to name, but copy it into the lambda
auto say_name_factory_copy(auto name) {
  return [=]() {
    std::cout << "Hello, " << name << "!" << std::endl;
  };
}

int main() {
  auto hello_jeff = say_name_factory_copy("jeff");
  hello_jeff();  // Hello, jeff!
}
```

However, copies can be expensive (e.g. vectors) or even impossible (think `unique_ptr`). Another option is to use move semantics:

```
// Say hello to name, but move it into the lambda
template <typename T>
auto say_name_factory_move(T &&name) {
  return [capture(std::forward<T>(name))]() {
    std::cout << "Hello, " << *capture << "!" << std::endl;
  };
}

int main() {
  auto jeff_ptr = std::make_unique<const char *>("jeff");
  auto hello_jeff = say_name_factory_move(
    std::move(jeff_ptr));
  hello_jeff();  // Hello, jeff!
}
```

Here, `name` is moved into `capture` inside the environment of the returned lambda. We use `std::forward` instead of `std::move` because `name` is a universal reference, but otherwise you can just treat this as a move.

Side note: This is exactly how move closures in Rust work. When you declare a Rust closure to be move, the closure takes ownership of any variables referenced inside its body. Of course, the fact that the borrow and lifetime checkers exist means that the compiler is very *explicit about when you must use a move closure to avoid dangling references. In other words, if the world used Rust instead of C++, I wouldn't have needed to write this article at all.*

However, moving lambdas comes with a catch: we can't use `std::function` to abstract their behaviour, because `std::function` takes a copy of the lambda[3]. If the captured variable is uncopyable (e.g. `unique_ptr`), then the lambda will also be uncopyable, and the compiler will give us a very nasty and undecipherable warning as C++ compilers are wont to do. This unfortunately means that our previous trick of using `std::function` to store lambdas in a container ceases to work. We've gained the ability to call C++ lambdas closures, but in turn we've lost the ability to call C++ lambdas first-class functions.

Thankfully, a solution to this problem comes courtesy of Raymond Chen from *The Old New Thing*[4,5] — we can simply use shared pointers for all captured arguments, and the function becomes copyable again:

```
// Say hello to name, but use a shared pointer
template <typename T>
auto say_name_factory_shared(T &&name) {
  // make a shared pointer to encapsulate name
  auto ptr = std::make_shared<T>(std::forward<T>(name));
  // capture copies ptr, thus extending its lifetime
  return [capture = ptr]() {
    std::cout << "Hello, " << **capture << "!"
              << std::endl;
  };
}

int main() {
  auto jeff_ptr = std::make_unique<const char *>("jeff");
  std::function<void()> hello_jeff =
    say_name_factory_shared(std::move(jeff_ptr));
  hello_jeff();  // Hello, jeff!
}
```

So there you have it. A way to turn any arbitrary C++ lambda into a first-class function with closure properties. This is C++ as jeff expected. This is C++ as the ISO working group intended. This is real, raw, down-to-the-metal C++, warts and pimples and all, Machiavellian as always, just as Bjarne Stroustrup envisioned it would be. No ifs, no buts, no small text or hidden terms and conditions. C++ is a functional programming language, and that title is here to stay.

Performance implications are left as an exercise for the reader.

terrifiED

If you want to play with the example code given, here's a Compiler Explorer link: https://godbolt.org/z/cPhzx4WYT

1. I got this very eloquent answer out of Prof. Richards at the end of CS 442. You should definitely take CS 442 if you have the chance.
2. The eagle-eyed may notice that `check_conds` doesn't have perfect forwarding. The fix is simple and is left as an exercise to the reader.
3. The reason that `std::function` has to take a copy of the lambda, rather than being a simple zero-cost abstraction/wrapper type, is well beyond the scope of this article. The gist, however, is that C++'s type system isn't powerful enough to properly represent functions and their environments, so the ISO C++ working group decided to force them to be copyable in `std::function`.
4. https://devblogs.microsoft.com/oldnewthing/?p=100635
5. I discovered Raymond's post after running into this exact problem in my CS 246E project — we had a lambda that was outliving its environment. This article is not an advertisement for courses designed by Brad Lushman, but it might as well be at this point.

DIRECTOR ADVERTISEMENT

No cartoons this issue, but there's an opportunity open next term (Spring 2022) for 1–2 directors for the MathSoc Cartoons project!

Are you interested in:

- Leading a team of writers/artists to create useful academic resources for math students?
- Developing valuable transferable skills?
- Being featured on the MathSoc website for your contributions?

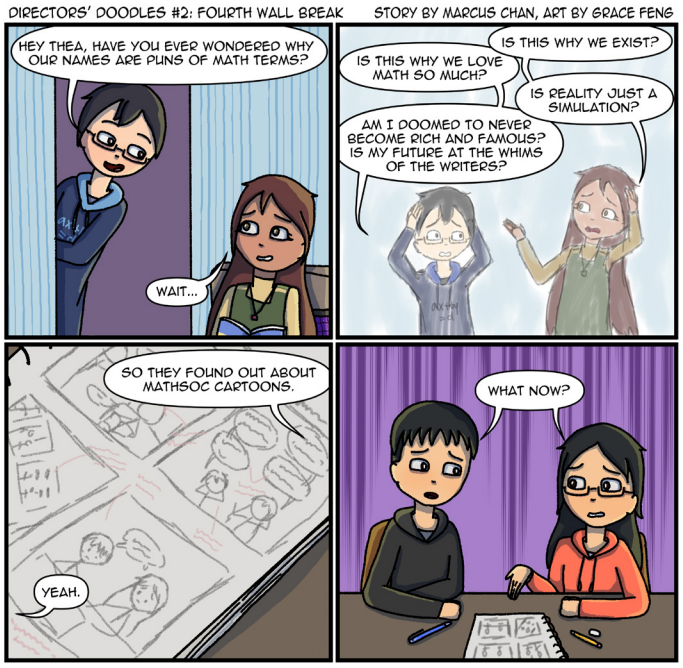
Then send your resume to cartoons@mathsoc.uwaterloo.ca now! **Deadline: Friday, March 4th, 11:59 PM EST.**

MathSoc Cartoons

DIRECTORS' DOODLES #2: FOURTH WALL BREAK

We're two cartoons in and already the characters have become aware of the fact that they're just that — cartoon characters. How do you deal with self-aware characters, and can someone please hand us another roll of duct tape to keep that fourth wall together?

More bonus content can be found on the MathSoc Discord server and on the MathSoc Cartoons socials.



MathSoc Cartoons Directors

ISSN 0705-0410
UW'S BASTION OF ERUDITE THOUGHT SINCE 1973

mathNEWS is a normally fortnightly publication, funded by and responsible to the undergraduate math students of the University of Waterloo, as represented by the Mathematics Society of the University of Waterloo, hereafter referred to as MathSoc. **mathNEWS** is editorially independent of MathSoc. Content is the responsibility of the **mathNEWS** editors; however, any opinions expressed herein are those of the authors and not necessarily those of MathSoc or **mathNEWS**. Current and back issues of **mathNEWS** are available electronically via the World Wide Web at https://mathnews.uwaterloo.ca. Send your correspondence to: **mathNEWS**, MC3030, University of Waterloo, 200 University Ave. W., Waterloo, Ontario, Canada, N2L 3G1, or to userid mathnews@gmail.com on the Internet.

mathNEWS is overseen by the Board of Publications, an autonomous board of the Federation of Students, University of Waterloo, hereafter referred to as Feds. **mathNEWS** is editorially independent of Feds and the Board of Publications. **mathNEWS** has never been requested to withhold Improper Content as defined under Feds Policy 71.

GRIDWORD

CALLING ALL GRIDWORD GAMERS

gridCOMMENT *148.3*

Hello to all the gamers!! So there I was in MC, wandering around waiting for class to start when I spot the last issue of **mathNEWS,** but low and behold, I flip to the back and see no **gridWORD**!!! I was what they might say, shaking in my boots (crying profusely), and would not stand for such crimes against humanity.

So now here we are! Call in the fire brigade cause new **gridWORD** just dropped and it's what they call **fire**. I cooked this one up special for you all, those who are craving some **gridWORD** action, enjoy it to your hearts content *:prayge:*

On another note, is it just me, or does everyone not see any geese on campus recently? I've seen a few, but none compared to previous terms spent at UW. So given this, I want to ask for this issues **gridQUESTION**, "Where did all the geese go?". This works well with this **gridWORD**'s theme as well, which is: ***GEESE!!!***

Can't wait to see your solutions and answers! If you have one, email it to mathnews@gmail.com by 6 pm on February 28th with your name or moniker, and include your **gridQUESTION** answer as well!

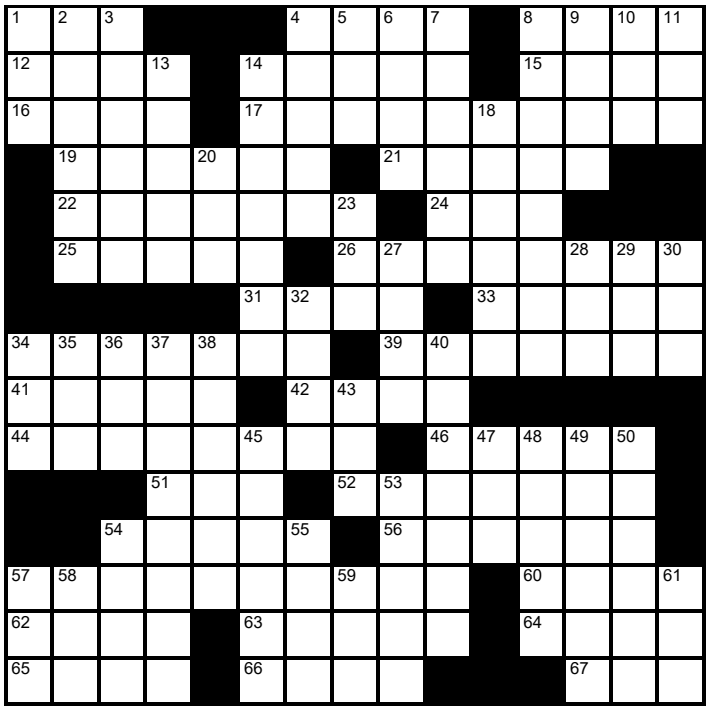
Wink wonk

ACROSS

1. Band aid?
4. No longer due
8. ___ cheese
12. Agreements to keep quiet
14. Main artery
15. Livin' la ____ Loca
16. A measurement for land
17. *What a spaniards neighbour may call a goose?
19. *What one might do to make an origami goose?
21. __ ___ on, meaning tedious length (2,3)
22. To trust someone enough to share private matters
24. Prefix with angle
25. _ ____ you so! (1,4)
26. As opposed to your first
31. Brother of Jacob
33. Expression of surprise or affirmation
34. A, in Morse code (3,4)
39. Ankle bones
41. A preface to a book or speech
42. __ __ them, said to one who must decide on a party to follow (2,2)
44. What a chef does when adding ingredients to soup (6,2)
46. Prohibited thing
51. It's as easy as ___!
52. What might happen if your work downsizes (4,3)
54. Menacing look
56. *What an American goose's favourite candy might be?
57. *What an English goose's favourite candy might be?
60. Almost a circle
62. "Dies ___", Latin hymn
63. Fancy tie
64. World's longest river
65. What an addition symbol might say?
66. Cat call
67. Similar to Inc.

DOWN

1. Santa ___, Calif.
2. 1801 in Roman numerals
3. Mimic
4. What one did in a bathroom
5. What a pirate might say
6. "Suffice __ __ say..." (2,2)
7. To make someone feel apprehensive
8. As opposed to mornings
9. Went out, as a fire
10. Commercials
11. Daisy ___
13. "I ___ __ issues here", what one might say when oblivious to the problem at hand (3,2)
14. The two furthest points of orbit, pl.
18. Belt at a wedding
20. Akin to the NFL, but European?
23. She, in Portuguese
27. Car
28. A small battery
29. Like the median, abr.
30. Serpentine letter
32. Feng ____, like furniture arrangement
34. Like math, science, arts, abr.
35. Gold, in Spanish
36. Commonly skipped on sign up, abr.
37. Removed salt from water
38. Single-celled creatures
40. What one might be if they show a strong display of artistic talent
43. A weekly snow on Saturday nights
45. A loud cry of anguish
47. Suffix to Gator-
48. Subatomic particle
49. Axis __ ____, like an alliance during WW2 (2,4)
50. A dash __ ____, to add flavour (2,4)
53. Bow projectile
54. Elated
55. Latin, to be
57. Nintendo avatar
58. "Oh my!", Japanese expression
59. Prefix with friendly
61. A common lightbulb type



Drop your gridWORD solutions off at MC 3030? 👉😟👈

A PERPETUALLY BORED mathNEWS EDITOR

lookAHEAD

SUN FEB 20	MON FEB 21	TUE FEB 22	WED FEB 23	THU FEB 24	FRI FEB 25	SAT FEB 26
Winter Olympics ends	Family Day	Employee Thank-You Day			Requests for final exam religious accommodations due	Tell A Fairy Tale Day

SUN FEB 27	MON FEB 28	TUE MAR 1	WED MAR 2	THU MAR 3	FRI MAR 4	SAT MAR 5
Reading week ends	Final examination relief requests due **mathNEWS** 148.4 Production Night	Application to Graduate due Mardi Gras (Fat Tuesday)	Cycle 2 interview period begins Ash Wednesday		National Salesperson Day **mathNEWS** 148.4 released	

MATHSOC SPRING 2022 ELECTIONS ARE NOW OPEN!

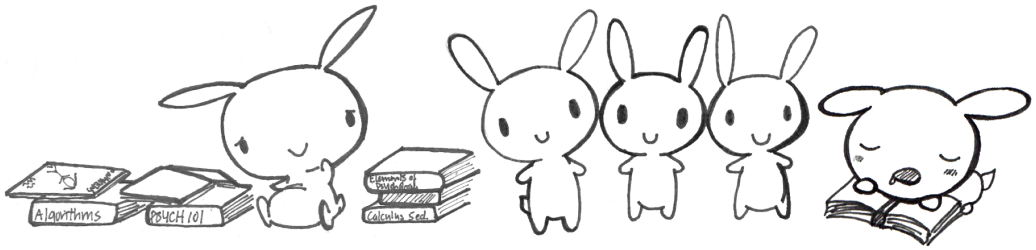
Are you looking to get involved with MathSoc and represent members of the society? Today's your day, because nominations for the MathSoc Spring 2022 General Elections are now open! Nominations close on Saturday, February 26th at 11:59 PM ET so make sure to get your nominations in ASAP! :)

Link for nominating yourself and endorsing others: https://vote.wusa.ca/elections

Want to learn more about Council? Head over here: https://mathsoc.uwaterloo.ca/council/

Got questions? Feel free to reach out to anyone on the Elections Committee – we're on the MathSoc Discord, accessible by the UW Discord Student Hub, or you can email us at elections@mathsoc.uwaterloo.ca.

MathSoc Elections Comittee
Winter 2022



otherNEWS is made technically possible by club executives of the Math Faculty.

I say "technically" because if they had sent us more news this week, this box wouldn't be here.

THE mathNEWS EDITOR WHO PUTS THE "NEWS" IN mathNEWS