βeta

— news from computer science and engineering —

Security is a Mindset

page 6

---

issue 130 — 22th May 2017 — 17s1 Week 12

βeta

CSESoc βeta, issue 130

A fortnightly publication from CSESoc's Beta team.
Find us online at **www.csesoc.unsw.edu.au**
Got some good content? Email
csesoc.beta.head@cse.unsw.edu.au

βeta Head
Melissa Zhang

The Article-ator Ben Pieters-Hawke
Puzzles Wrangler Kevin Tran

Contributors
Gary Bai Dan Kennedy Thomas Flynn
Adam Smallhorn

In This Issue

Editorial

Welcome to the last issue of Beta for this Semester!

This week's exciting installments include the latest news, upcoming events, and a look into the world of security. Remember to read the weekly Soc Announce and regularly check out our facebook page at *https://www.facebook.com/groups/csesoc/* to stay up to date with the ins and outs of CSESoc!

Want to be a part of the team? Beta is always welcome to new members, so just find us on Facebook at **CSESoc Beta Team 2017** or send me an email at *csesoc.beta.head@cse.unsw.edu.au*. We'll help you find your place within the team where you can sit back and watch your ideas come to life in delectable black-and-white print!

Have a wonderful Winter break and I'll see you next Semester!

■ *Melissa Zhang*

What's On

Today! CSESoc's Weekly Barbecue social

12:30–2pm, John Lions Garden

Come on down to John Lions Garden for your weekly dose of free barbecue! Don't forget to pick up your copy of CSESoc βeta, and make some new friends!

29 May Awesome BBQ x Bakeoff social

12:30–3pm, John Lions Garden

CSESoc is bringing you the biggest and best BBQ of the entire semester - THE AWESOME BBQ!!!

In addition to our regular sausages we will have special meats, salads and snacks to feed you well over extended hours so that you can keep your stomachs full until the next semesterly BBQ on Wednesday Week 1, Semester 2. Bring your friends, its going to be HUGE!

Additionally we are also holding this event in parallel with our annual CSESoc Bake Off competition! This is the perfect chance for you to show off and bake something for our members at the BBQ and win some AMAZING prizes along the lines of last year's CSESoc Apron and embroidered wooden spoon! Cakes, cupcakes, biscuits, or even your own zesty invention - we want to try it! Don't want to fork out your own money? We'll be offering $10 Woolworths vouchers to the first 5 registrations. For more information, check out the event on Facebook!

Tech Roundup

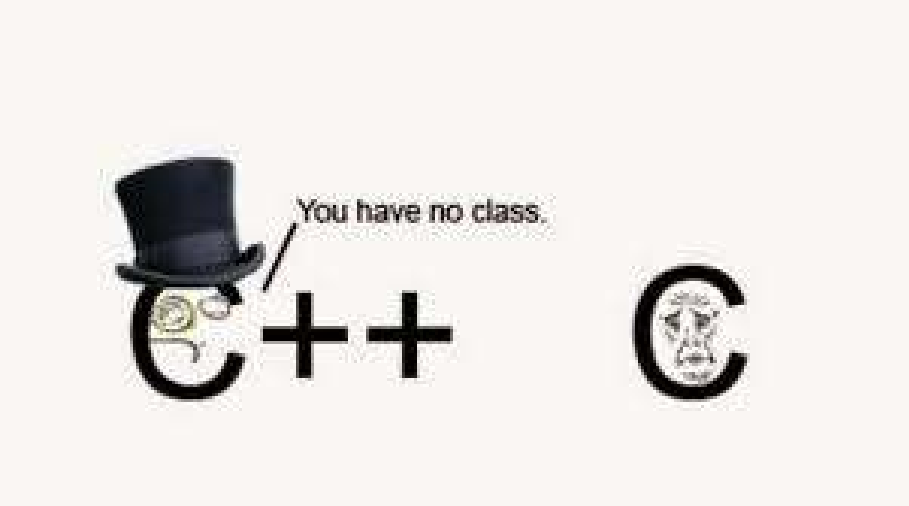
Welcome to Tech Roundup!

Google I/O, Google's annual developer conference, was held last week near the company's Mountain View headquarters. There were a number of announcements and products shown off - too many for this short piece. We'll focus on what was arguably the most popular and exciting one - Google Lens, a cool new image recognition technology. Google Lens can recognize what a phone's camera is seeing, and perform an appropriate action - such as connecting to a wi-fi router, translating text, finding reviews, and so on. There isn't a solid release date for the Lens yet, but it's likely the technology will be integrated into pre existing apps such as Photos, or Assistant.

Not a fan of Java, but want to do Android development? There's plenty of ways to get around that problem - Xamarin, React Native, and so on - but have you ever considered Kotlin? Perhaps you have - it's been possible for a while - but Android Studio 3.0 is going to include official support for Kotlin. And unlike Go, the support for Kotlin will be first class - which means Kotlin will support the full Android API. In related news, Google and JetBrains will also be collaborating to create a non profit foundation for Kotlin.

Speaking of Developer conferences, Microsoft also held their annual Build conference last week. Perhaps the most relevant piece of news for those of you using Windows is the announcement of the "Fall Creator's Update", another major Windows 10 update, which will arrive sometime in the Australian Summer. Besides the design, store, and Cortana changes to be shipped with this update, readers might also be interested in the improvements Microsoft will be making to the Linux Subsystem - the major announcement regarding this topic being the inclusion of other Linux distributions, as well as USB device communications, allowing you to use the subsystem for things like Raspberry Pi.

■ *Gary Bai*



Canva Frontend Engineering talks – 31 May 2017

Canva will be hosting their second Frontend Engineering talks on May 31st. The main speaker will be Ryan Cavanaugh, one of the core contributors to TypeScript at Microsoft. Cavanaugh says:

"TypeScript started when it became apparent at Microsoft that we'd be writing many more large JavaScript applications in the future, but JS wasn't well-suited for large-scale application development due to maintainability problems presented by a lack of static typing…. TypeScript inspires me because it solves a real problem in a simple, elegant way. There's nothing I love more than seeing someone on Twitter say something like "I was hesitant at first, I'm never going back to JS after using TypeScript"

Ryan, Sharon Kuo and Marc Fallows of Canva, as well as Chloe Chen of the CSIRO will be talking about "typed JavaScript" technologies and what the future holds for them. So grab a ticket (free delicious dinner and drinks included) and come and mingle with frontend engineers from all over Sydney!

Where: Canva office, 2 Lacey Street Surry Hills

When: May 31, 6:30-8:30pm

Tickets: $10 on Eventbrite. Free UNSW tickets are sold out, paid tickets selling fast!

■ *Dan Kennedy*

Artisans in the Editor

Style is regarded as a fundamental property of good code. Generally, a programmer's first introduction to style is through an organisationally-mandated 'style guide' that details everything from where to place one's braces to which language-features not to use. Curiously, the concepts of style and taste extend to many areas within computer science (CS) and software engineering (SE). I sat down with senior lecturer Dr Eric Martin of CSE to delve into the basis of style and its reach through the disciplines.

I began by posing the question - To what extent can we afford to pay attention to style, when other factors - being efficient, being productive, being cost-effective - seem to be of much higher importance?

Dr Martin explained how reading and writing code can be looked at allegorically as reading and writing literature. When we begin to read a work of literature, we become aware of the author's way of expressing themself - their use of particular language, how sentences are constructed and the resulting flow of the story. As one observes their writing - 10 pages in, 20 pages in - we get to know the patterns and predict their choices. After some time we start to think as the author thinks. Their style becomes familiar and importantly, the patterns of their writing unsurprising.

He goes on to suggest that this allegory is employed as a model of collaboration between programmers. We use style as norms - a set of guidelines that a group adheres to. This reduces the chance of the 'authors' creating surprises. Style as norms allows a programmer to step into a project and become productive as an author of the story. This is the case of style in the realm of organisations, deadlines and productivity.

Alternatively, we can address style as elegance. Dr Martin references the French naturalist and mathematician, Buffon - "The style is the man" [0]. He elaborates further - "Being a human being is first having a particular style - that is what defines you". When we look to solve a problem with code, when do we consider that we are finished? Does the work end on the basis our program passes all test cases? According to Dr Martin, the real work begins after we find a solution. The process of polishing is when a programmer transitions from engineer to artisan, crafting programs for which every line is considered over and over again in pursuit of clear, succinct, powerful statements. From such a meticulous approach yields code which is personally satisfying to the programmer. With this satisfaction, ingrained in the files, in the source, is the essence of this person - their style.

I then considered the broader reach of style into CS. Dr Martin stated in a lecture on object-oriented programming that "Within computer science there are many churches". How does this affect a programmer's approach to solving problems - is a style enforced by each paradigm individually, or is it a 'church' in and of itself (as preached in Clean Code [1], for example)?

Dr Martin says the crux of this statement can be likened to a user of a particular operating system. They begin fumbling their way through until eventually mastering its environment. They transition to a power-user. It is at this stage, they see solutions to their problems in the tools and features of this particular operating system. To think outside this system is impossible until an alternative is learned. I reflect on this analogy with the cliché describing confirmation bias – "If all you have is a hammer, everything looks like a nail". Dr Martin makes it clear how followers of paradigms may champion them as a perfect solution to writing code, the extent to which philosophical arguments are made for their absolutism.

Ultimately, this outlook exposes how personal taste can influence the higher level decisions of solving a problem. In practice, as programmers and engineers, we should look at the tradeoffs between various approaches to solving problems and collect them in our tool box. Dr Martin shared his own preference to language selection:

"Within programming languages, we have lots of syntactic variations to write particular statements. A language which is very rich in syntactic constructs makes it harder in some ways… That is why I like lean languages, where you have pure syntactic constructs to choose from. Then you can play this game of: Can I find the best possible style?… We are then putting the emphasis on choice of algorithm than on choice of how to translate this algorithm into a piece of code."

I then turned to the pragmatic goal of writing software – creating systems to solve problems. In The Art of Unix Programming [2] the notion of developing an "intuitive feel for the Unix style" is emphasised. Do systems themselves embody the style of the code they are comprised of?

Dr Martin opens with the classic phrase "Small is beautiful." As a system becomes more complex - made of more and more modules – we typically look at it from an increasingly abstracted point of view to make decisions about its architecture. This necessity to manage the codebase from a higher level perspective tends to flatten out notions of style and elegance, relating back to how norms function as a collaborative model between teams of programmers.

This brings us back to the way we looked at style in different contexts and the way in which we incorporate its notion in our own work.

He draws focus to his approach as a mathematician "I like to polish mathematical proofs and definitions as much as I like to polish code." He feels that an elegant solution is one that is unique by its definition - often we find that multiple occurrences of a solution can be shown as equivalent under certain assumptions. We have no reason to hold one above the other. When we provide some additional constraints or

alter the assumptions these break down, and we conclude that we have missed something within the problem. Similarly, when writing code, we must be able to determine when one solution is 'more correct' than another. It is a common situation to write what seems like a functional solution only to find edge cases that immediately break it, in which case we must change our thinking (or simply add a 14th nested if statement).

I wanted to know, as programmers, if we could consider a pursuit of more stylish code as a means to improving technical ability.

Pointing out the pressures of balancing work, university and life, Dr Martin admits that it can be very difficult for programmers to find time to properly consider style in their code. One way we can try to consider style is from the time we think about the problem, design a solution, to the time we code it and finally, test it. However, if we relate back to the notion of the real work happening after we find a solution – reflecting on what is written and really thinking hard about if we can make it simpler, leaner, more elegant – Dr Martin says, "Forcing yourself to ask these questions, you can really begin to take style seriously." A way in which we can assist this process is by writing good comments – explaining a piece of code to yourself in the clearest possible manner often suggests a way to improve your code. Dr Martin states that often he makes changes to code just after writing a comment. Conclusively, all this deep work on style occurs after we have found a solution. It requires discipline to go back and work on what is already an acceptable solution, which is usually a luxury under the constraints students and professionals alike are faced with.

Dr Martin sums up the interview with his thoughts on why it is important to pursue the notion of style outside of merely the appearance of code, "It makes programming more enjoyable, giving you a sense of fulfilment – as you become more satisfied you become more passionate, more motivated and keener to push yourself harder and harder. As a side effect, you will achieve more."

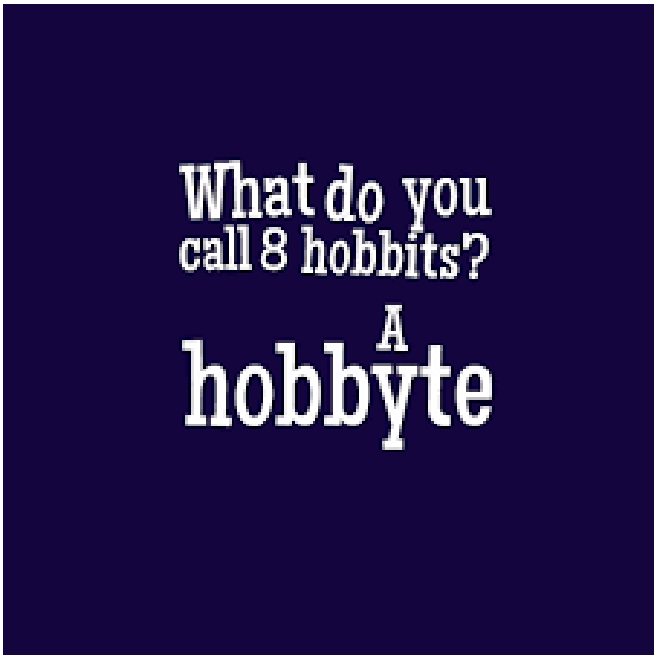
By looking past the goals of simply getting our programs running, having the 'correct' appearance and passing all test cases, we can allow ourselves to experience a deeper satisfaction in writing software.

[0] Discours sur le style (the "Discourse on Style"), George-Louis Leclerc De Buffon, 1753.

[1] Clean Code, Martin, R.C., 2008.

[2] The Art of Unix Programming, Raymond, E.R., 2003.

■ *Thomas Flynn*



Security is a Mindset

To understand the importance of security in today's world, you only need to look at the 1983 film "Wargames" - a film about nuclear missile control vulnerable to hacking. In 1983, if you wanted to make a film about the dangers of hacking, you needed to convince the audience that hacking computers could actually cause real-world damage. In 2017 nobody needs convincing - our news is filled with the latest attacks; from businesses to hospitals all the way to the failed North Korean Missile tests.

Security breaches are becoming increasingly problematic. In 1983 there was not much damage you could do by hacking a server, but today vast chunks of our critical infrastructure is connected to the internet - our hospitals, our power grids, our financial networks. We now also collect more personalised data about ourselves than ever before and we give connected devices unprecedented access to our lives. The threats to our economy, privacy and safety are clear, but we've only recently realised the more subtle influence poor security practices can have on fundamental parts of our democracy like our elections. All hackers seek these things : data, control and influence. Unfortunately the situation is likely to get worse before it gets better.

The solution to the security problem is not a firewall; it's not an app, machine learning or even just writing secure code - but rather the development of a mindset. At UNSW we have the philosophy that to be a good defender you must understand the techniques of a good attacker. We ought to know, having secured 1st place in CySCA - a Federally sponsored hacking competition for university students - every year since it's inception, including as recently as last week.

However this 'hacking mentality' is not some deviant ideology out of Kensington - it is the prevailing thought process for the entire security industry. Hacking is baked into the culture of security engineers across the globe.

Many years ago when starting university I was given a piece of advice from a retiring engineer - "I wish there had been someone to tell me where the industry was going, what new challenges and issues were going to be important in the decades after I graduated". For this decade - the challenge we face is security. The security of our future is a problem for all of our generation, but the solution lies in the hands of few - people like you with the technical ability to make a difference.

So how can you get involved? What role can you play? I think it's a mortal sin for any computer scientist to leave university without a grasp of how to write secure, defensive code, so if you can, enrol in a security course before you graduate. Many security courses, including UNSWs, are online and just a quick online search away. But don't stop there, investigate for yourself; do a little security research on the side. You're at the best university in Australia for Cyber Cyber, so take advantage of it. Learn what it is like from the other side, Get involved with a Capture the Flag competition (better known as a CTF) where you can test your hacking skills in a simulated environment. Join the UNSW Security Society and leverage the strong community we have.

We need a generation of critically thinking engineers with a trained security mindset before anything will get better.

■ *Adam Smallhorn*

Easy CTF

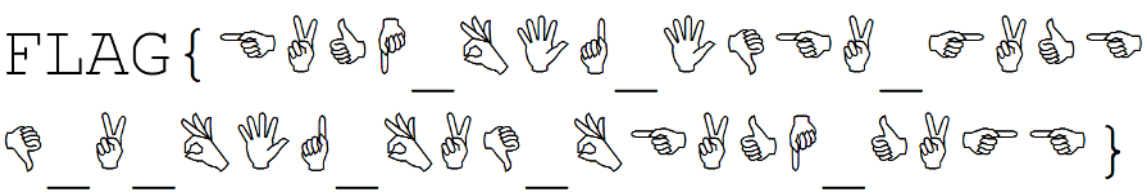
Want to get into Capture The Flag competitions but don't know where to start? Try EasyCTF! It's a competition pitched at people who don't have any experience in CTFs (or in security at all).

You can find the problems and solutions here: *https://writeups.easyctf.com/*

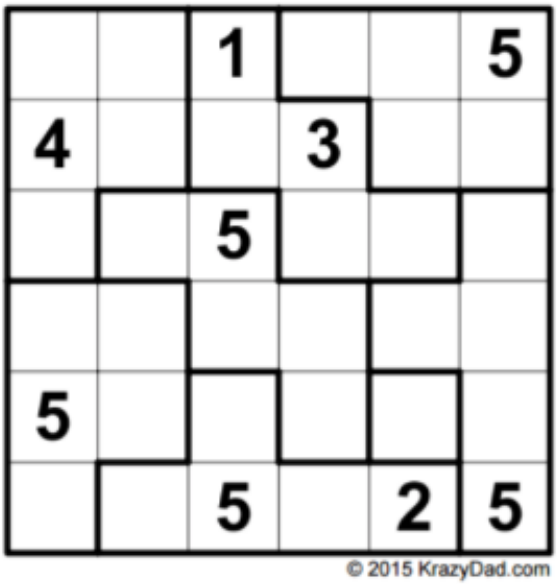
Here's two of the problems:

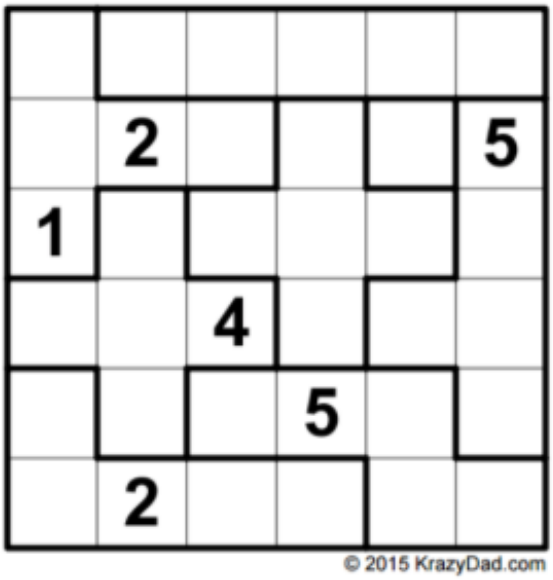
Please help me write the Fizz Buzz program (which reads in an integer N on standard input and prints the numbers 1 to N, each on a new line, replacing multiples of 3 with Fizz, multiples of 5 with Buzz and multiples of 15 with FizzBuzz).

Oh no! Two of my keys are broken! Can you write the program without '?' and 'i'?



Logic Puzzles





Suguru

The heavy lines indicate areas, called cages, from one to five squares in size. Fill each cage with unique digits, counting up from 1. For example a 2-square cage contains the numbers 1 and 2; and a 5-square cage contains the numbers from 1 to 5. Adjacent (touching) squares, even ones that touch diagonally, may never contain the same number

Brain Teasers

Puzzle #1

There are three kids. The teacher has six treats: three chocolate bars and three lollipops. The teacher will secretly hand each kid two treats at random so that they cannot see what the other kids got. You would like to know what treats the first child got, but you may only ask him one question, to which he can only respond with: "yes" "no" "I don't know"

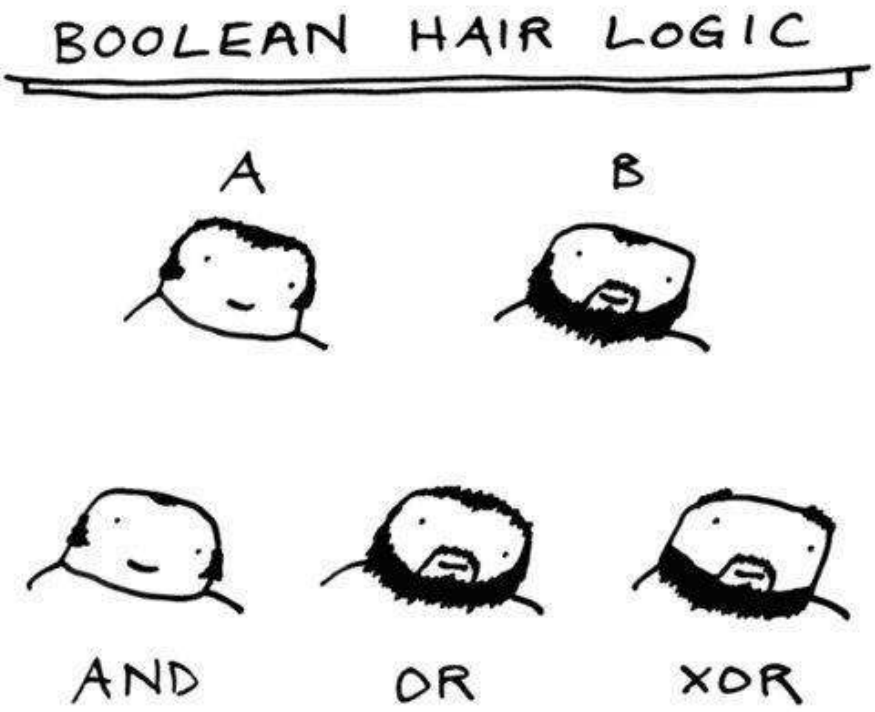
What question do you ask? You may assume the kids are, of course, perfect logicians.

Puzzle #2

Everyone knows that tic-tac-toe is a draw, but what about loser's tic-tac-toe? That is, you win if your opponent makes three in a row. Should you play first or second, or is this game also a draw?

```
["Hip" , "Hip"]
```

Hip Hip Array



This Edition of βeta Sponsored By...

UNSW AUSTRALIA
Computer Science & Engineering
Faculty of Engineering
atomi
Google
FORMalytics
freelancer.com
WiseTech GLOBAL
ARC CLUBS SUPPORTED BY Arc INDEPENDENTLY RUN
facebook
Jane Street
Microsoft
Atlassian
Palantir
Canva