

Program Reasoning

1. Introduction

Kihong Heo

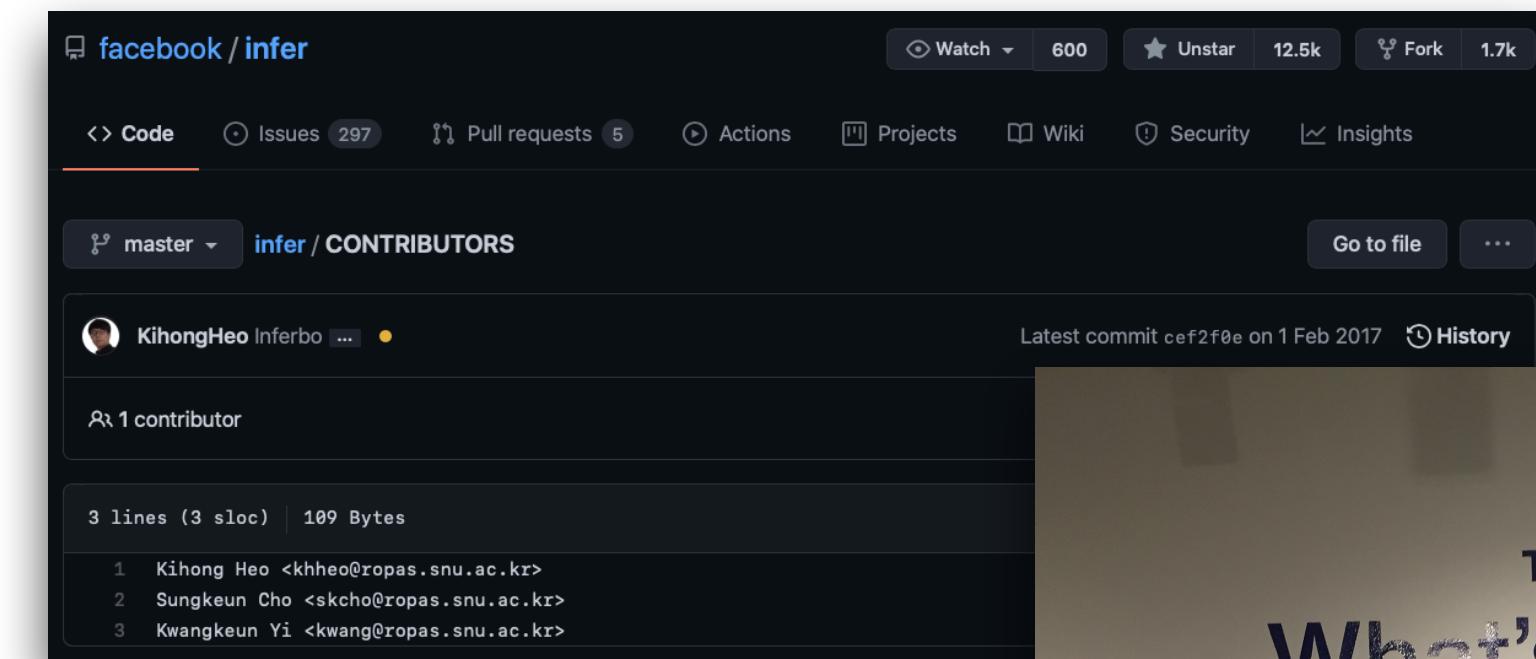


About Me

- Instructor: Kihong Heo (허기홍, kihong.heo@kaist.ac.kr)
- KAIST CS / GSIS / Programming Systems Lab.
- Homepage: <https://kihongheo.kaist.ac.kr> / <https://prosys.kaist.ac.kr>
- Office: N5 2321
- Office Hours: after each class (by appointment)

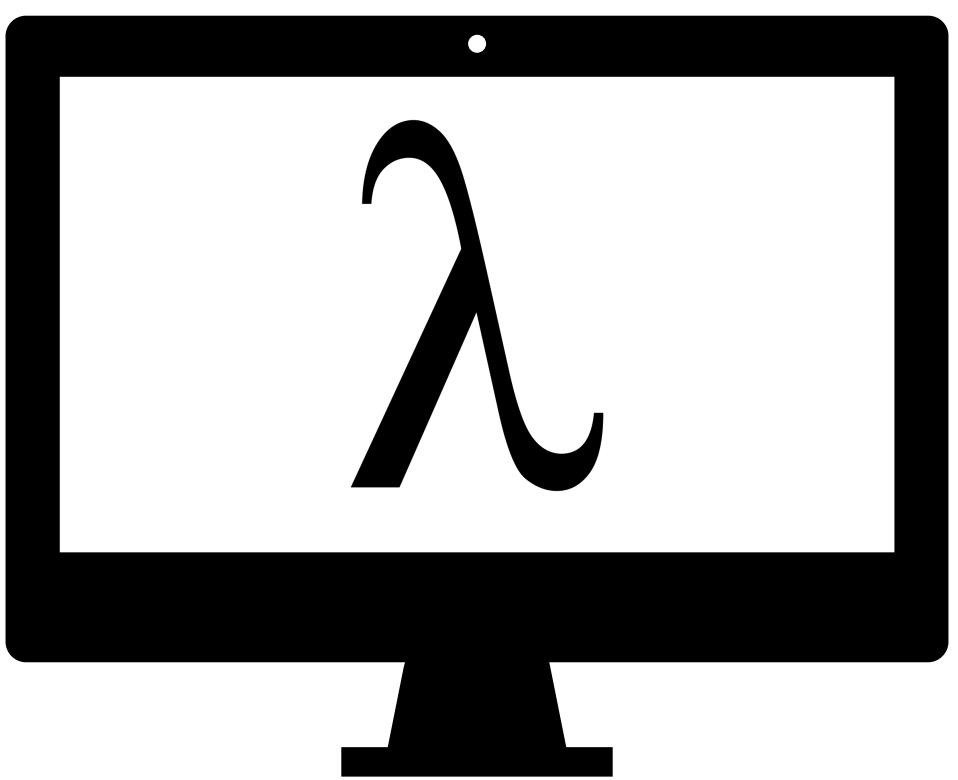
My Research

- Goal: solid PL theories \Leftrightarrow powerful programming systems
- Keywords: programming language, program analysis, SW engineering, SW security
- Good memories:



*<https://research.fb.com/blog/2017/02/inferbo-infer-based-buffer-overrun-analyzer/>

My Research

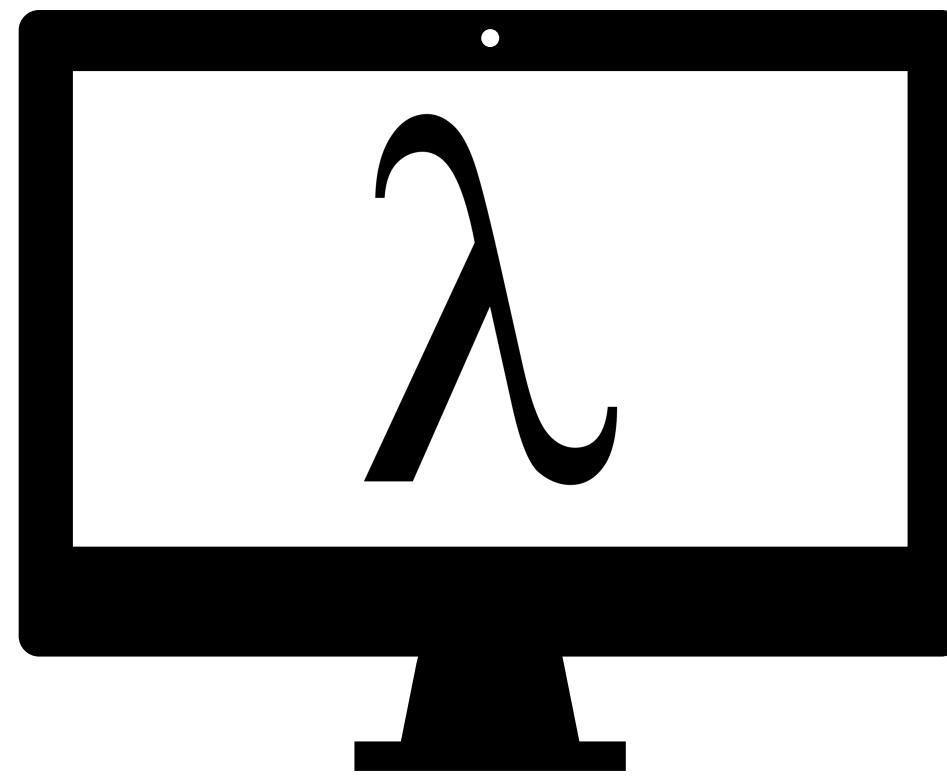


**Next-generation
Programming Systems**

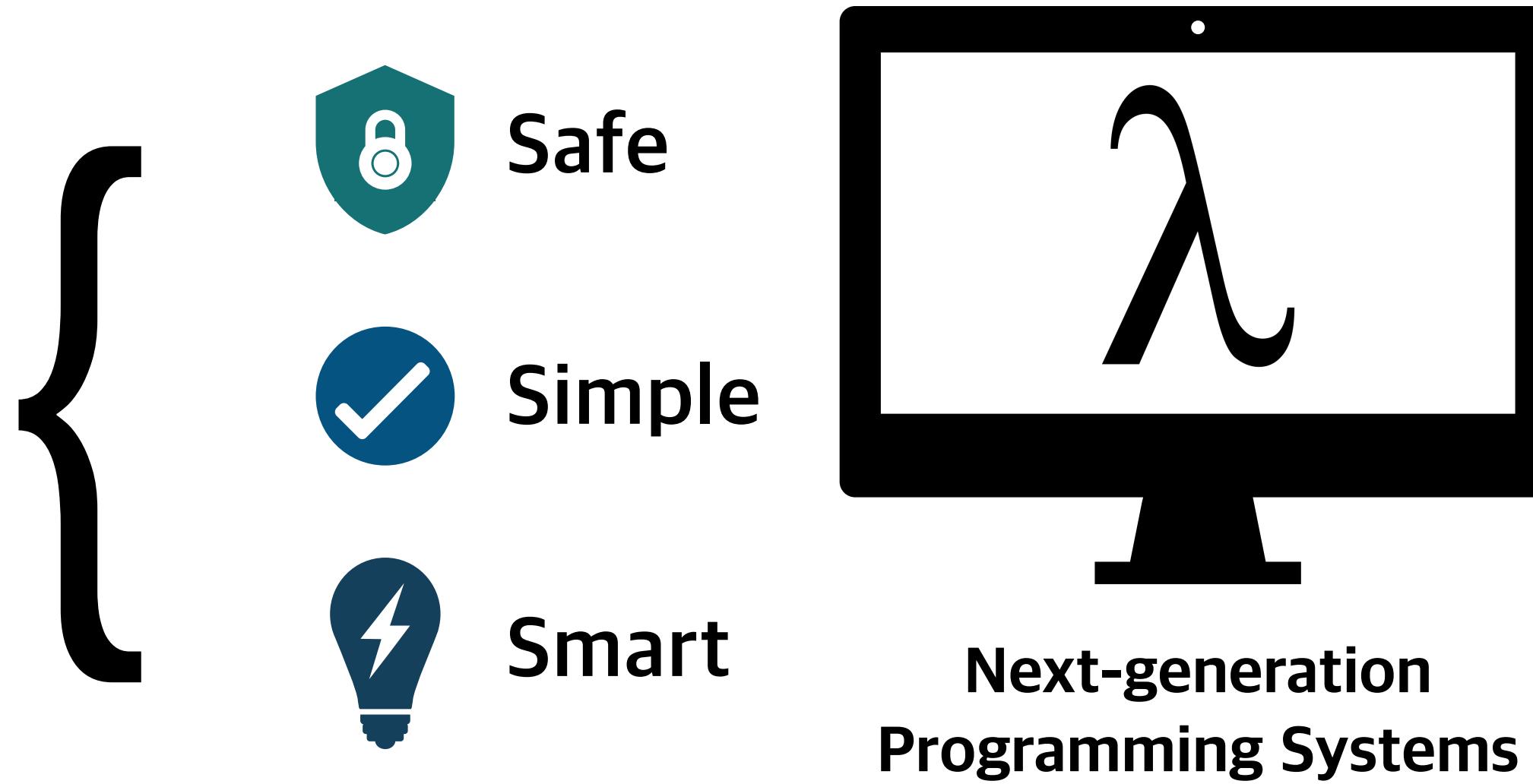
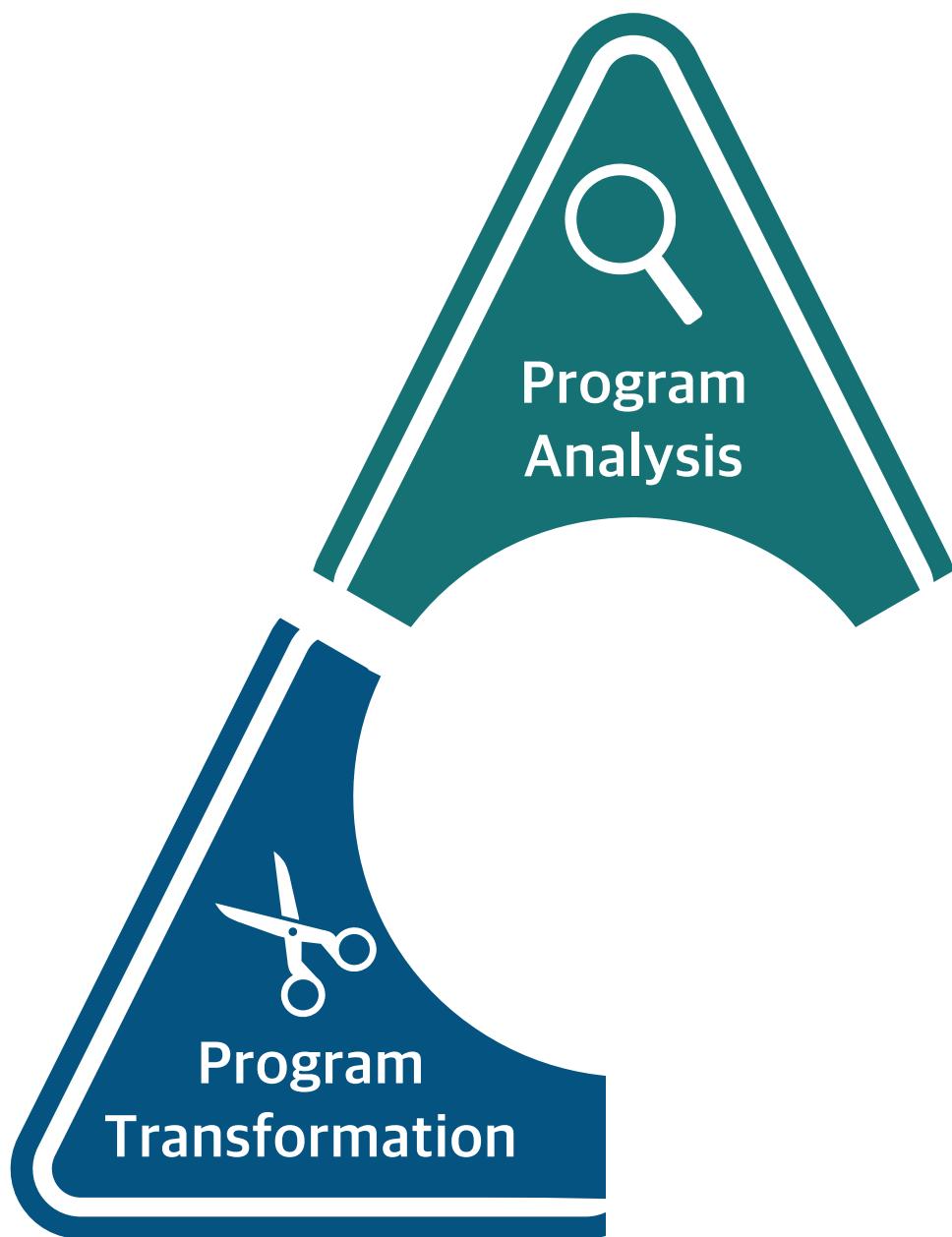
My Research



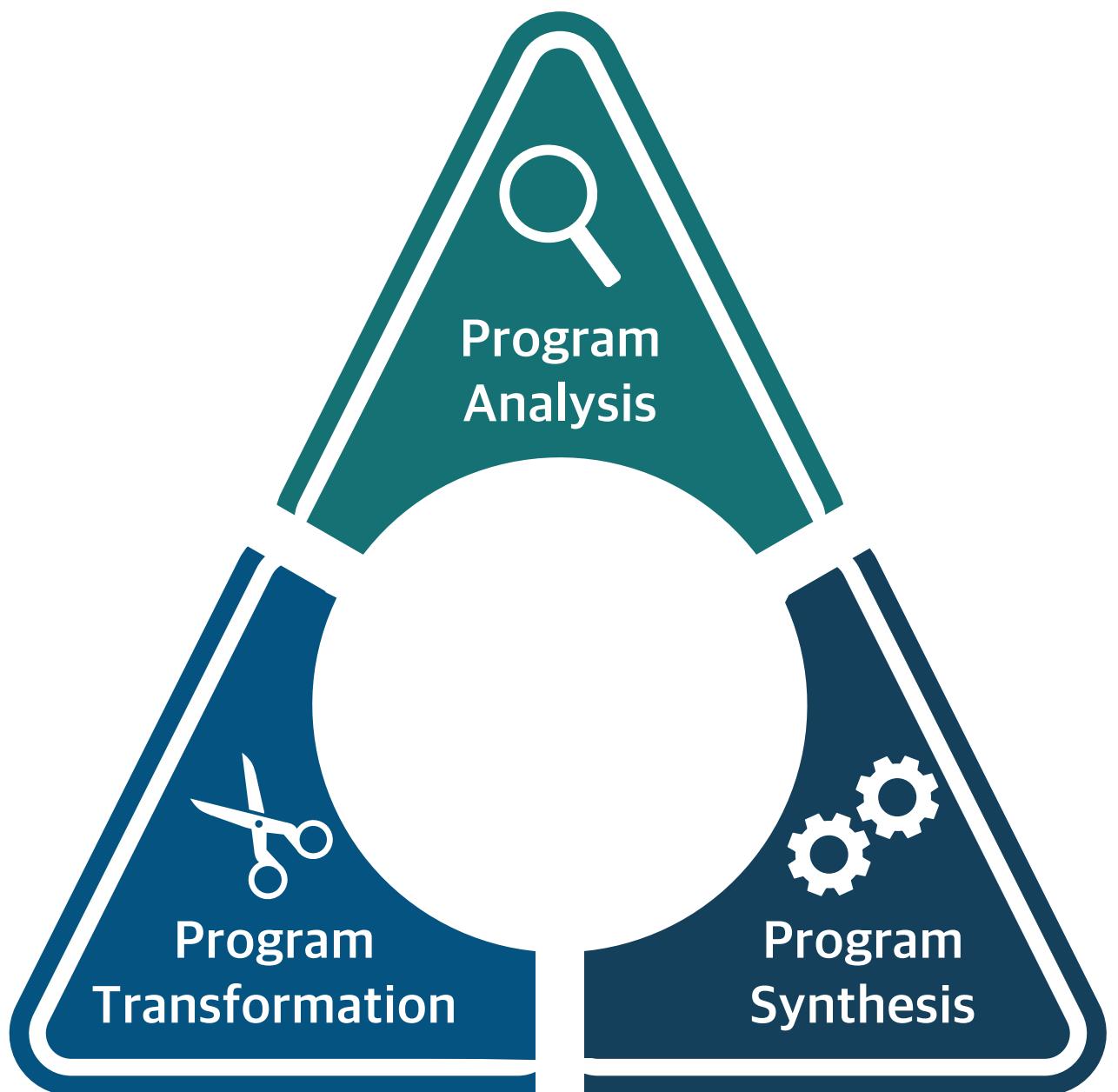
{

**Safe****Simple****Smart****Next-generation
Programming Systems**

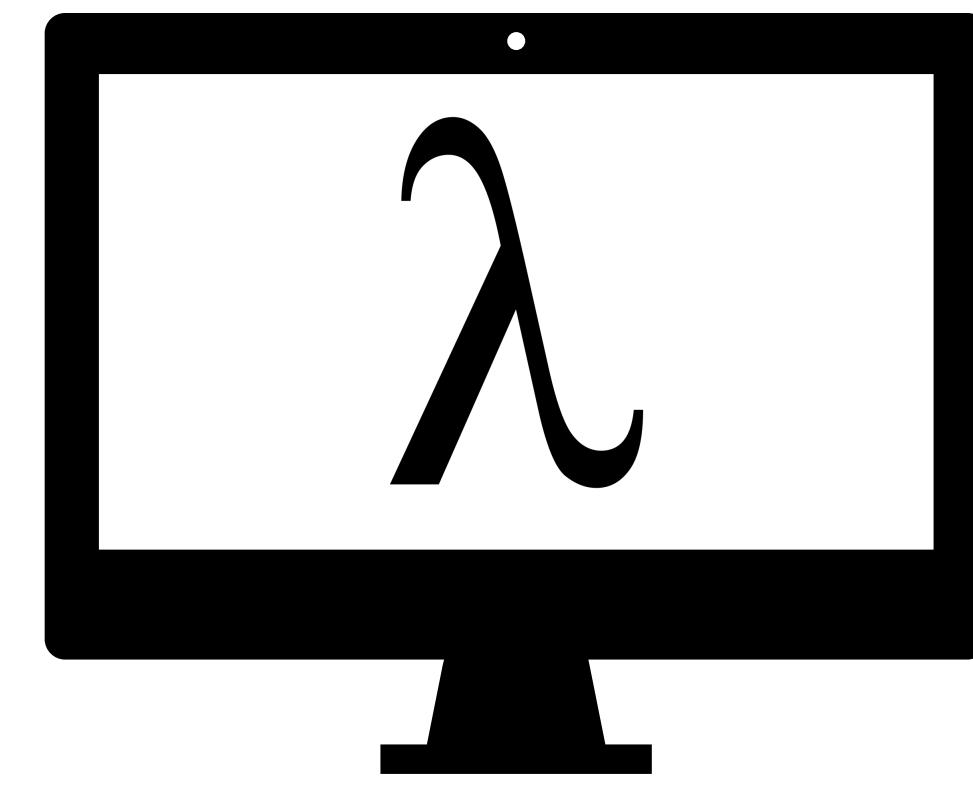
My Research



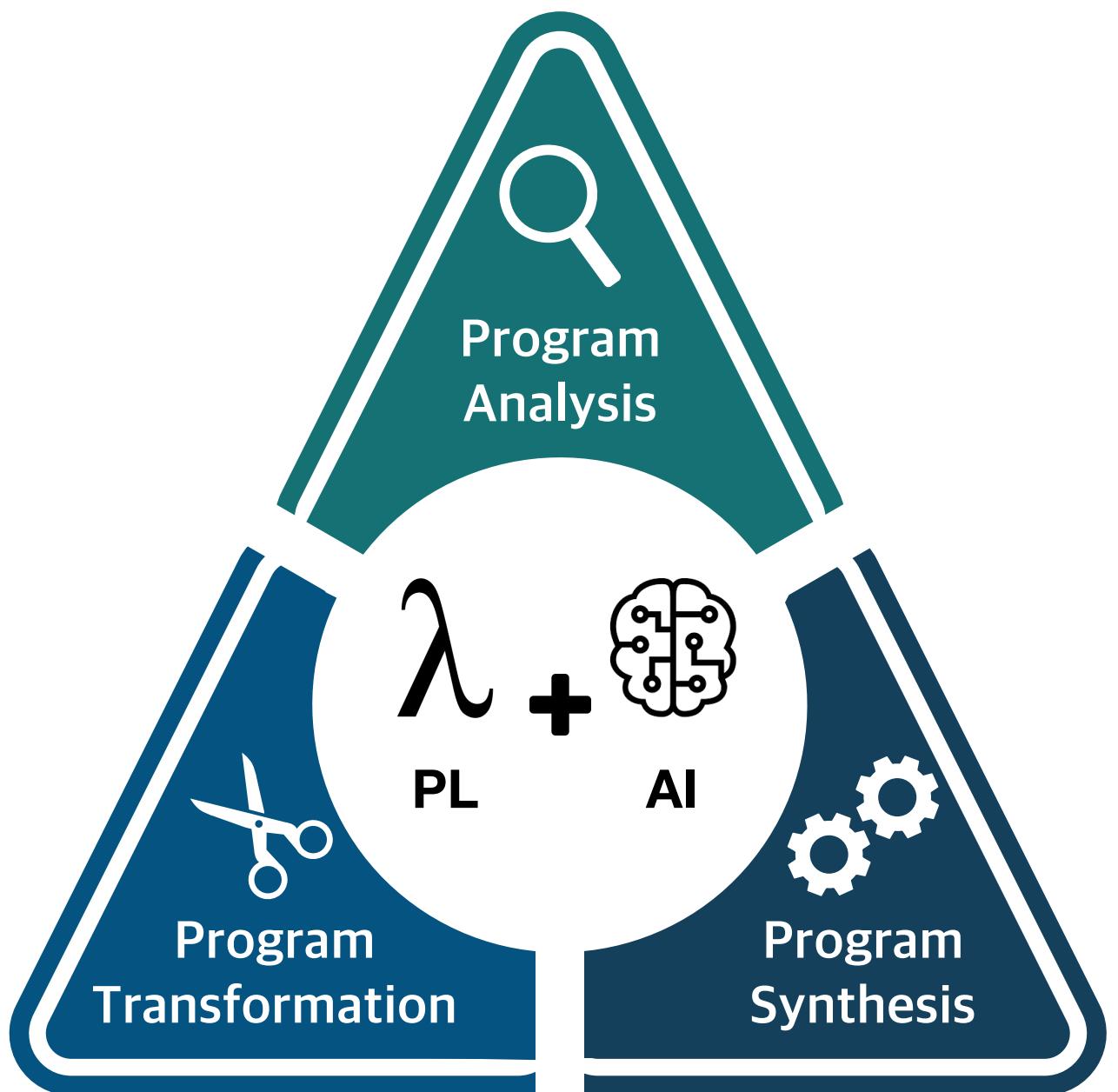
My Research



{

**Safe****Simple****Smart****Next-generation
Programming Systems**

My Research



{



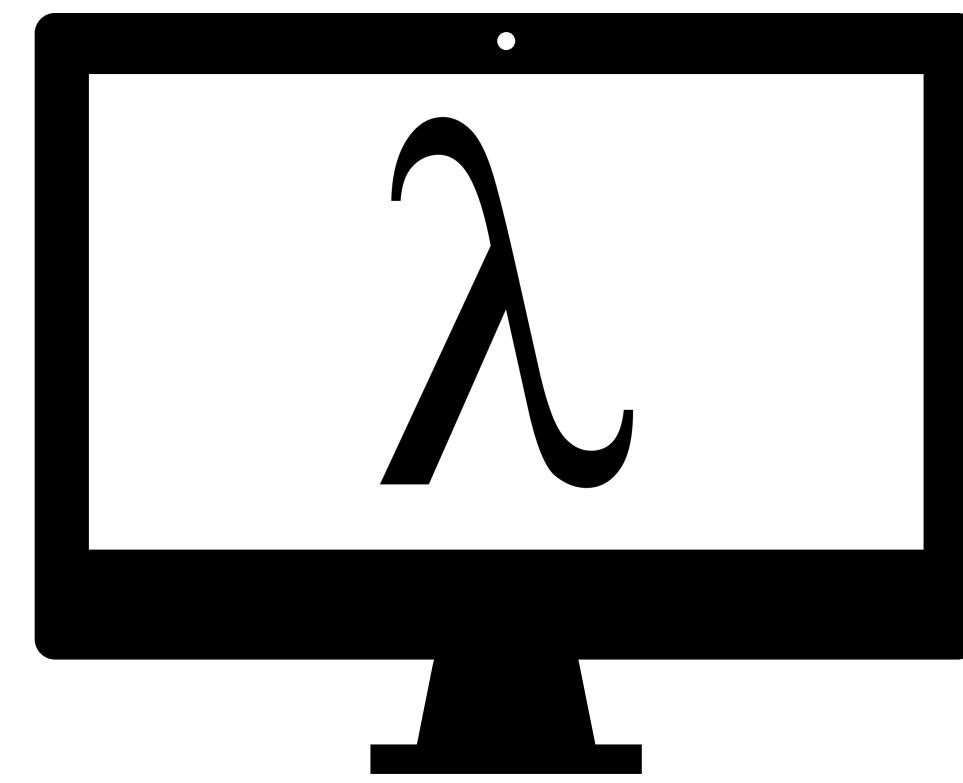
Safe



Simple



Smart



Next-generation
Programming Systems

Course Information

- Course Website: <https://github.com/prosyslab-classroom/cs492-program-reasoning>
- Q&A Board: <https://github.com/prosyslab-classroom/cs492-program-reasoning/issues>
- TAs (mailing list: cs492.ta@prosys.kaist.ac.kr)
 - Tae Eun Kim (김태은, taeeun.kim@prosys.kaist.ac.kr)
- Textbook:
 - Lecture slides will be provided
 - See the course webpage

Grading

- Homework: 50%
- Final exam: 40%
- Participation: 10%
 - Active participation including questions or discussions (online or offline)
- DO NOT allow for P/NR grading
- NOTE: **nonnegotiable!**
 - DO NOT send an email for a negotiation (cheating)

Important Notice (1): Academic Integrity

- DO NOT share the course contents (e.g., assignments or exams) with others
 - Esp., Github public repository, chegg.com, etc
- DO NOT discuss the details of solutions with others
- DO NOT plagiarize
 - We keep all the submissions from previous years!
- Any integrity violation: at **LEAST F**
- If you have questions: QnA board > TAs > instructor

전산학부 명예규정 2022 봄학기 / School of Computing Honor Code 2022 Spring

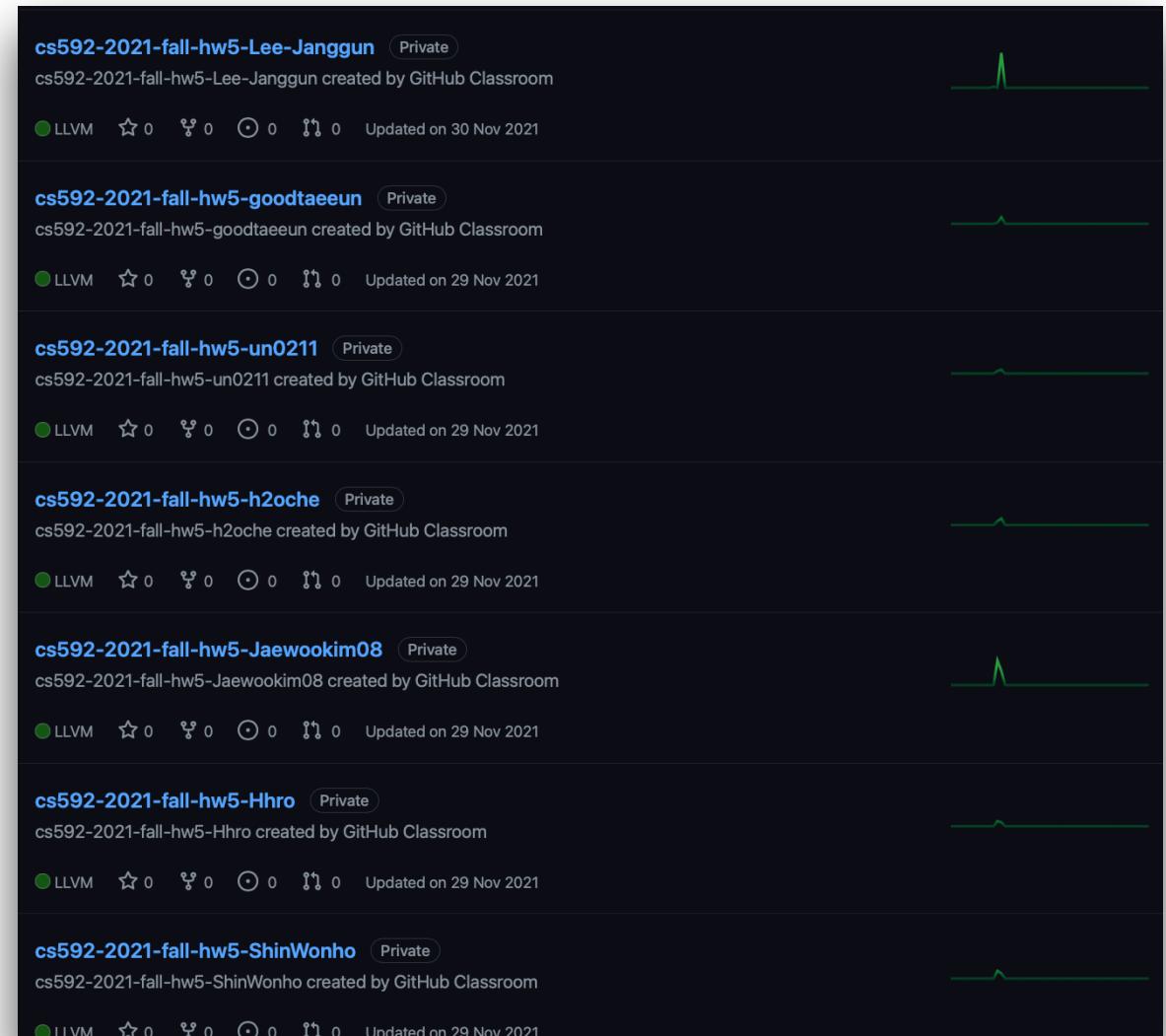
카이스트 전산학부가 운영하는 모든 수업에 참여하는 학생은 개인의 명예와 타인의 권리를 함께 존중하며 성실성과 정직성을 지키기 위하여 최선을 다합니다. 모든 시험 및 과제를 작성에 있어 허위되거나 다른 형태의 도움도 받지 않습니다. 다음의 행위들은 학생의 성실성과 정직성을 위반하는 것으로 간주됩니다:

- 본인 이외의 사람/기관이 작성한 답안지, 숙제, 프로그램 소스 코드, 보고서 등을 참고 및 이용하는 행위
- 시험 및 과제들과 관련해 [chegg.com](#)과 같이 정답을 공유하는 온라인 서비스를 이용하는 행위
- GitHub 등의 코드 저장소에 본인의 과제 답안을 공개하거나, 타인이 공개된 답안을 참고 및 이용하는 행위
- 다른 학생이 본인이 작성한 답안지, 숙제, 프로그램 소스 코드, 보고서 등을 참고하도록 용인하는 행위
- 다른 학생이 작성한 결과물을 자신의 것인 양 제출하는 행위
- 다른 학생을 대신해 시험을 치루는 행위
- 개인이 수행하도록 되어있는 take-home 시험이나 과제를 작성에 있어 허락 없이 공동 작업을 하거나 부적절한 도움을 받는 행위
- 표절, 길이와 무관하게 적절한 인용이나 언급 없이 타인의 창작물(참고서적, 문헌, 온라인상의 자료)을 무단으로 사용하는 행위

규정 위반 여부의 판단과 처벌 수위는 교과목 담당 교수에 의해 결정됩니다. 모든 부정행위는 전산학부 학사주임 교수 및 학부장님께 보고되어, 적발시 전산학부 내부적으로 아래와 같은 제재를 받습니다.

- 항후 2학기 동안 모든 포상 및 장학금 수여/추천 대상에서 제외함
- (주진공이 전산학부가 아닌 경우) 항후 2학기 동안 전산학부로의 전과 금지

위반의 실각성에 따라 학교 전체의 상벌위원회에 회부될 수 있습니다. 카이스트 학사규정이 적용하는 칭계의 범위는 아래 첨부된 학생 징계 일정 기준을 참고하세요 (학생 핸드북 한글판 67페이지, 다운로드는 https://portal.kaist.ac.kr/ennotice/student_notice/11614934649338)

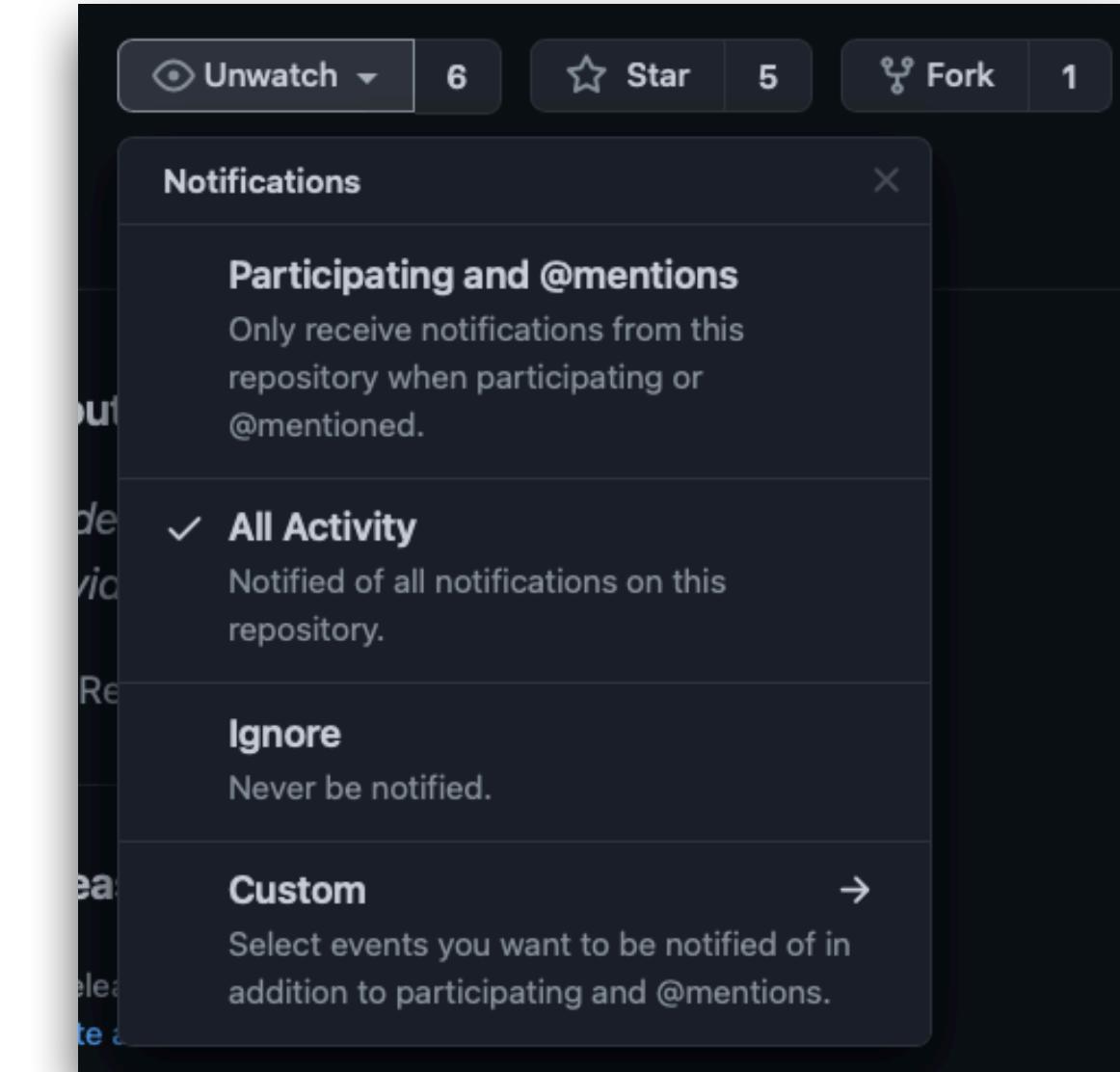


Important Notice (2): In-class Operations

- Language: English (default), Korean (supplementary)
- Attendance: always (default), absence (if necessary)
 - No quantified attendance score
 - “What is **essential** is **invisible** to the eye” - The Little Prince
 - I expect you to be here, as you expect me to be here!
- Questions & discussion (either in Korean or in English): highly encouraged

Important Notice (3): Out-of-class

- All Q&A and public notices: Github issue board
 - “Watch” all notifications
- Private notices (grading, etc): KLMS
- Questions are always welcome except for
 - Too detailed ones (TAs are not debuggers!)
 - Directly related to the solutions
- Actively discuss with your classmates



A long time ago
in a galaxy far, far away....

**SOFTWARE
BUGS**

Software Bugs: A Persistent Problem

- A long time ago, far far away



The Patriot Missile (1991)
Floating-point roundoff
28 soldiers died



The Ariane-5 Rocket (1996)
Integer Overflow
\$100M



NASA's Mars Climate Orbiter (1999)
Meters-Inches Miscalculation
\$125M

- Unfortunately, it becomes your own problem now

CNN U.S. | World | Politics | Money | Opinion | Health | Entertainment | Tech | Style | Travel | Sports | Video | Live TV | **US**

The 'Heartbleed' security flaw that affects most of the Internet

By Heather Kelly, CNN
Updated 5:11 PM ET, Wed April 9, 2014

A large red heart icon with a white outline, symbolizing the Heartbleed bug.

This dangerous Android security bug could let anyone hack your phone camera

By Anthony Spadafra November 23, 2019

Camera app vulnerabilities allow attackers to remotely take photos, record video and spy on users

A smartphone displaying a green binary code pattern over a keyboard, illustrating a security vulnerability.

AERIAN MARSHALL / TRANSPORTATION 06:30 2019 07:00 AM

What Boeing's 737 MAX Has to Do With Cars: Software

Investigators believe faulty software contributed to two fatal crashes. A newly discovered fault will likely keep the 737 MAX grounded until the fall.

A Boeing 737 MAX airplane flying through a cloudy sky.

Homeland Security warns that certain heart devices can be hacked



New in Life & Style

Interfaith 4th-graders bond through poetry, art and Steph Curry 2:03 PM

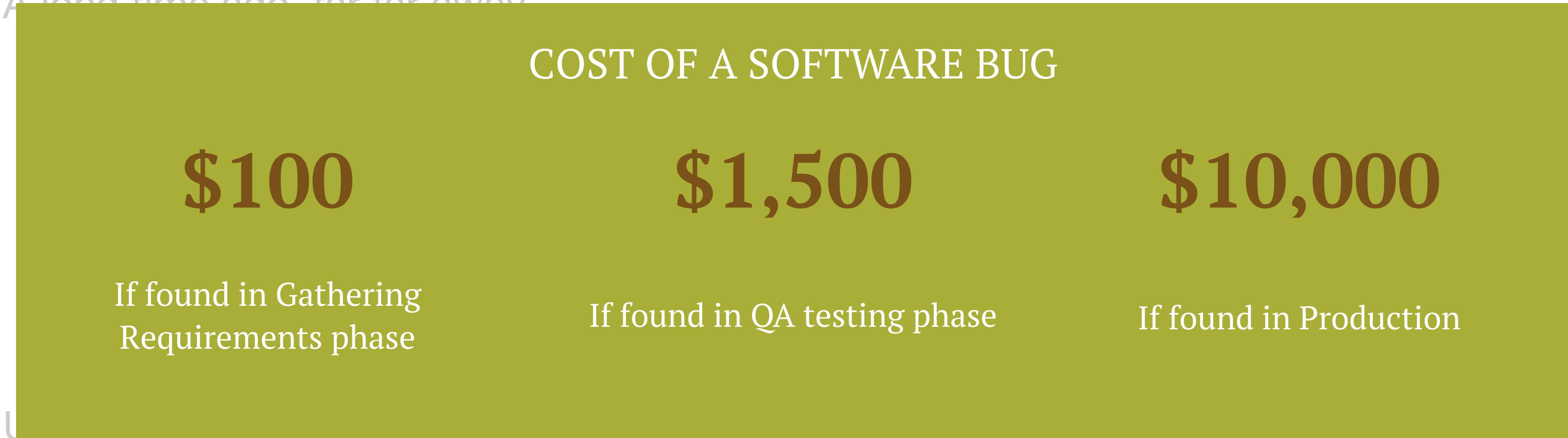
6 ways to celebrate Valentine's Day in Lake Geneva 8:55 AM

Six ways to keep your kids healthy during winter 8:56 AM

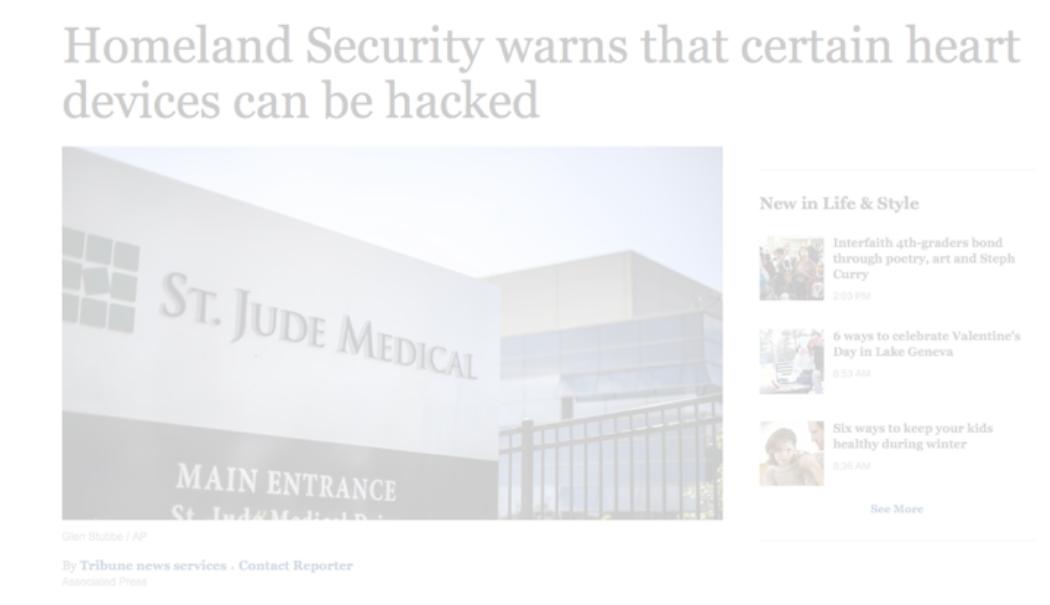
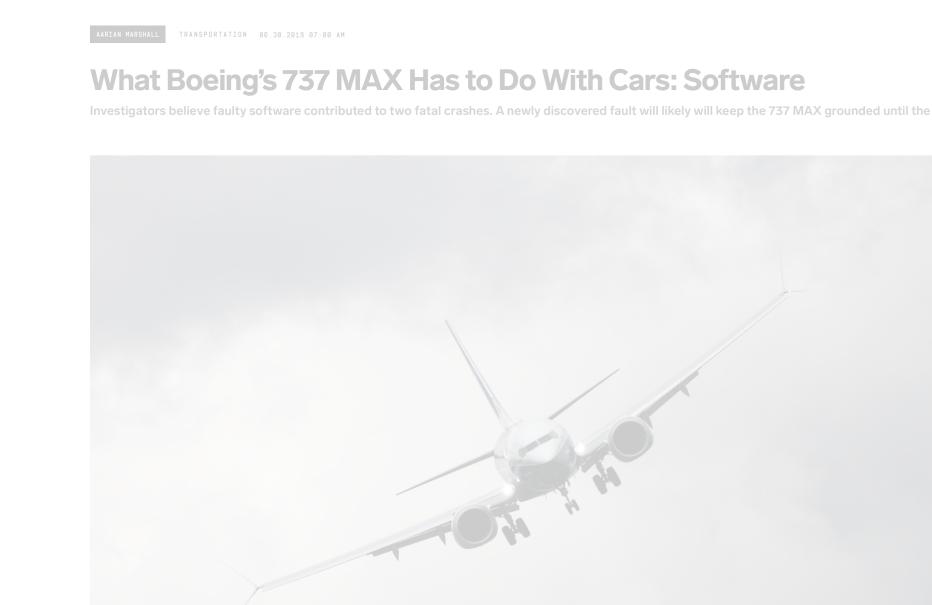
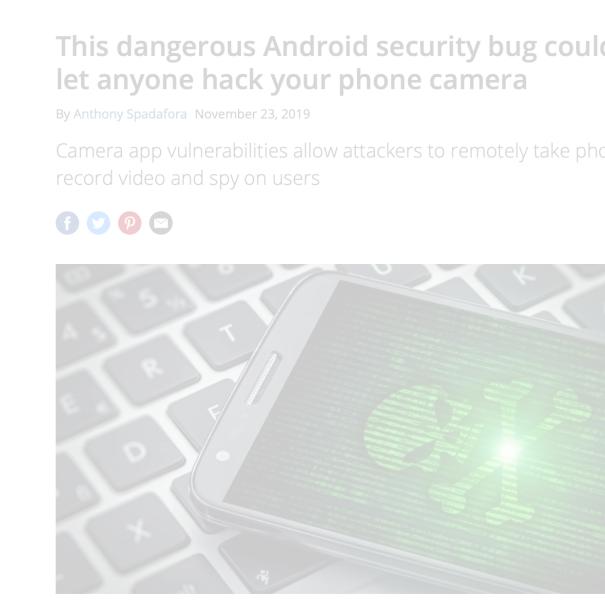
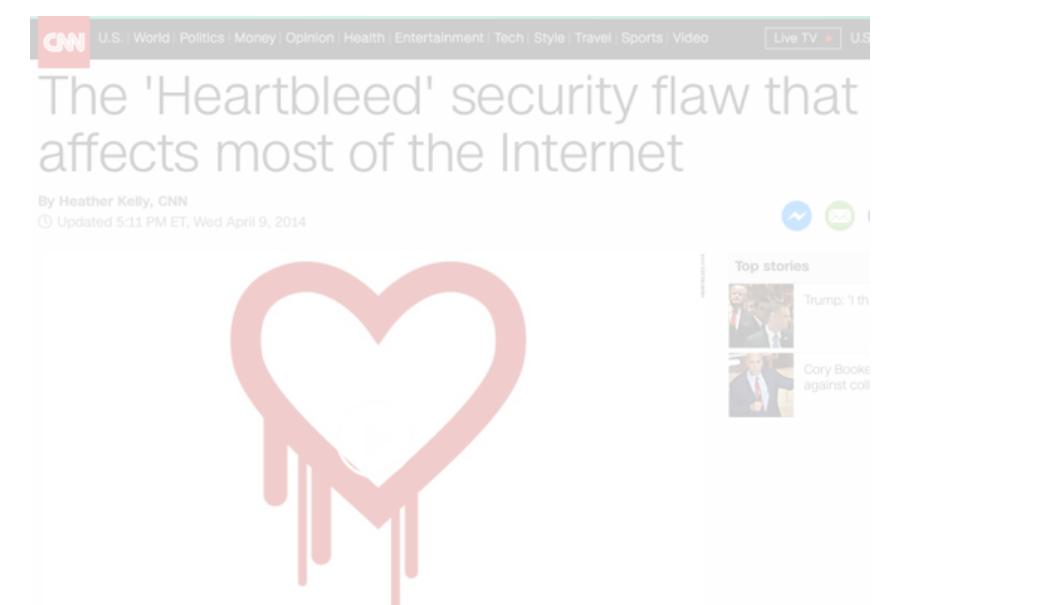
See More

Software Bugs: A Persistent Problem

- A long time ago, far far away...

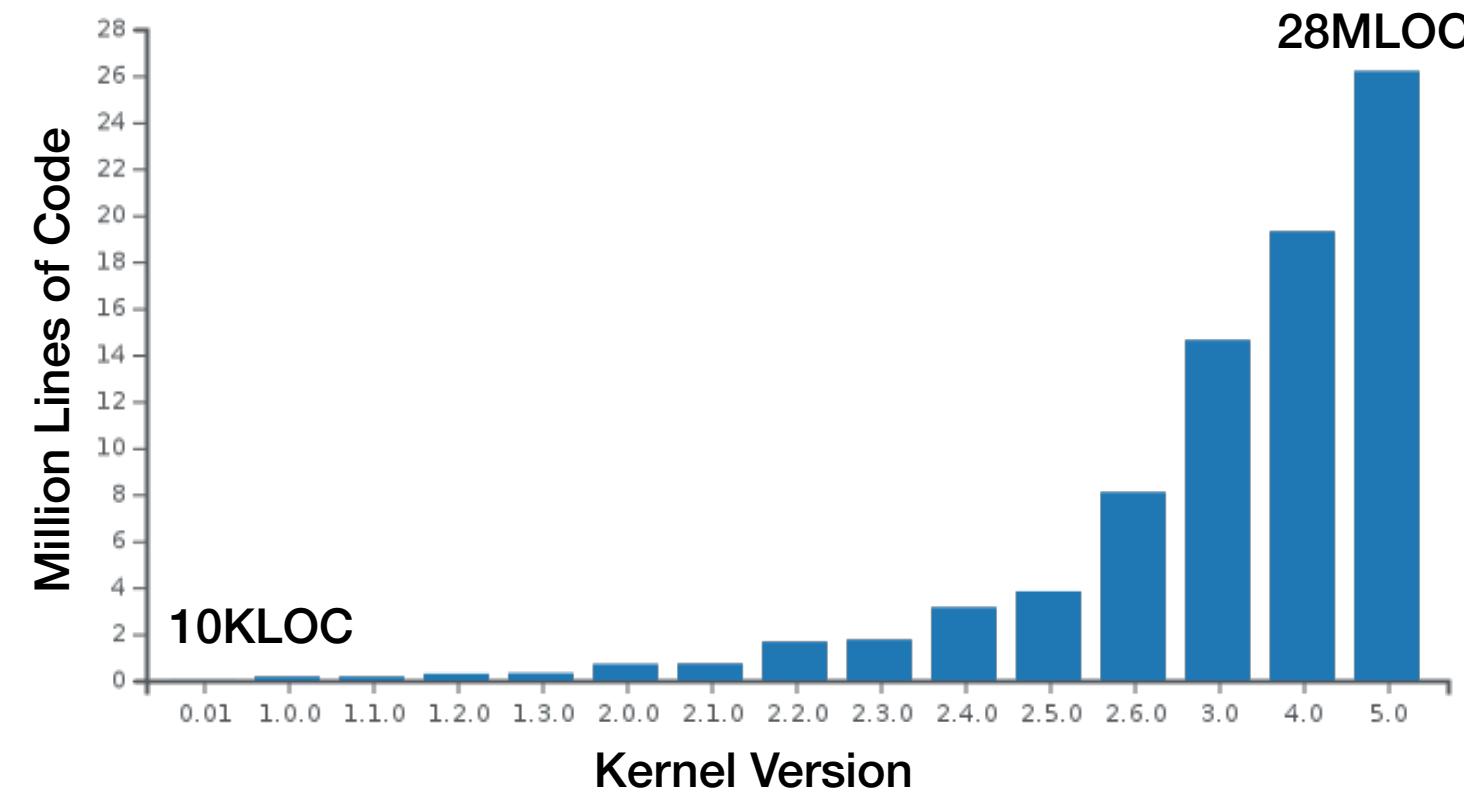


- IBM Systems Sciences Institute, 2015

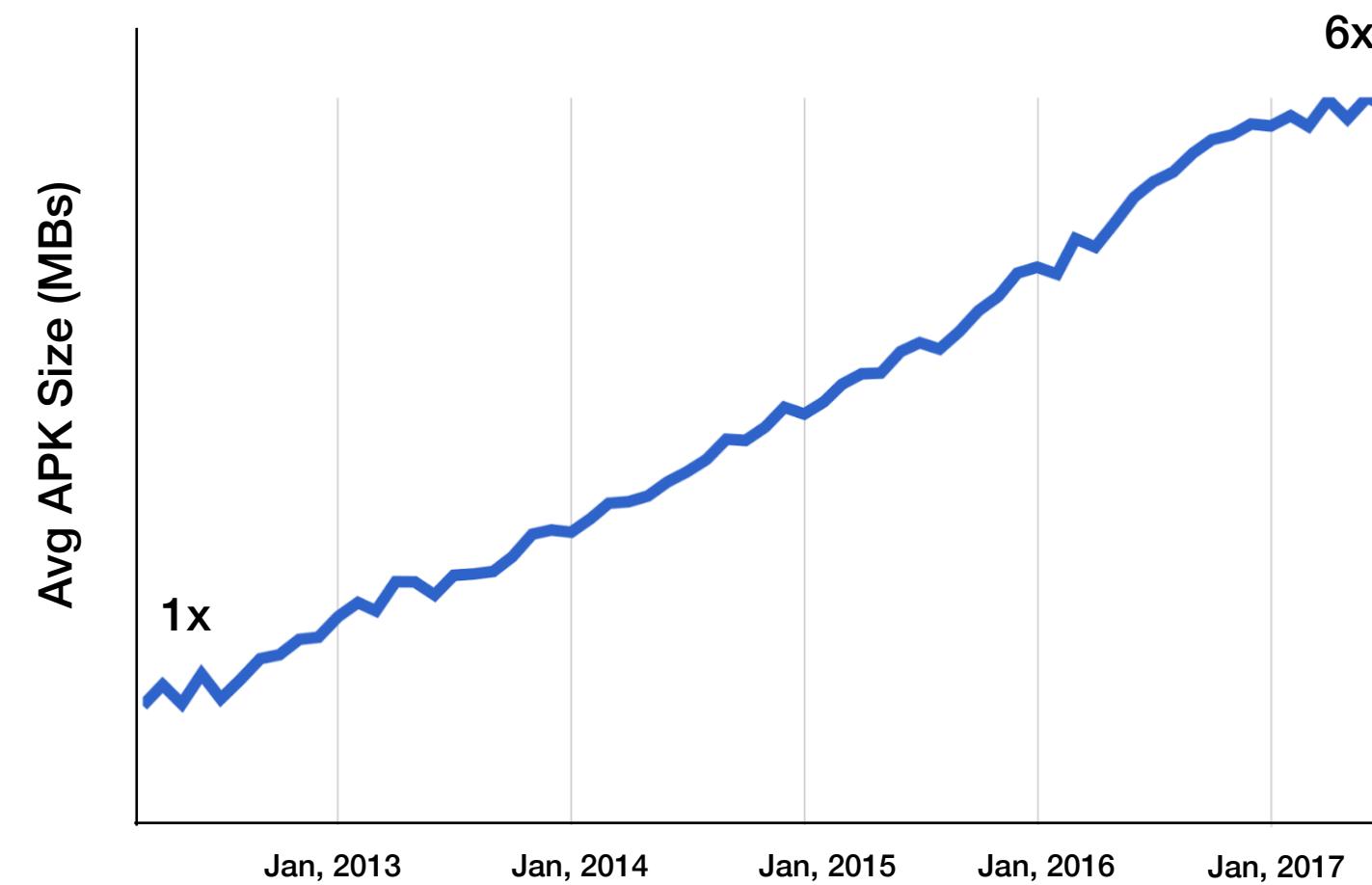


Why Software Still Fails?

Size of Linux Kernel



Avg. Size of Android Apps



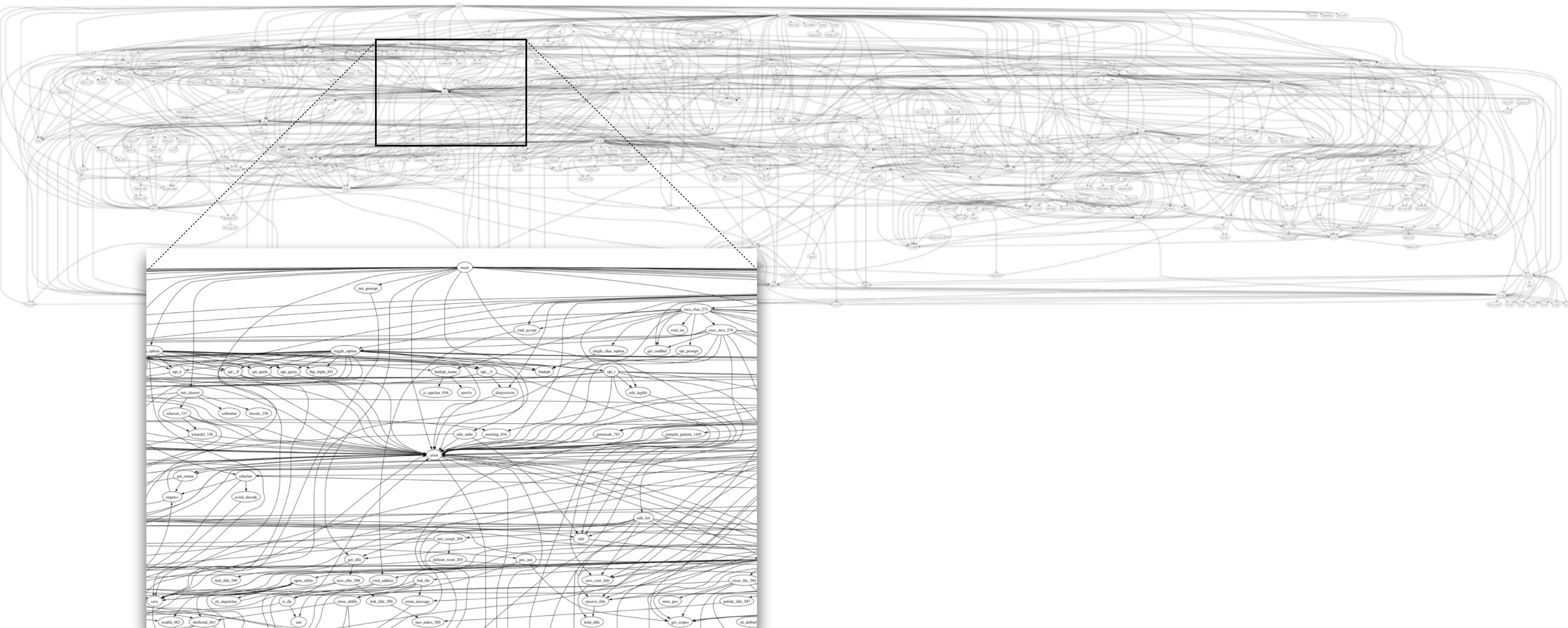
X



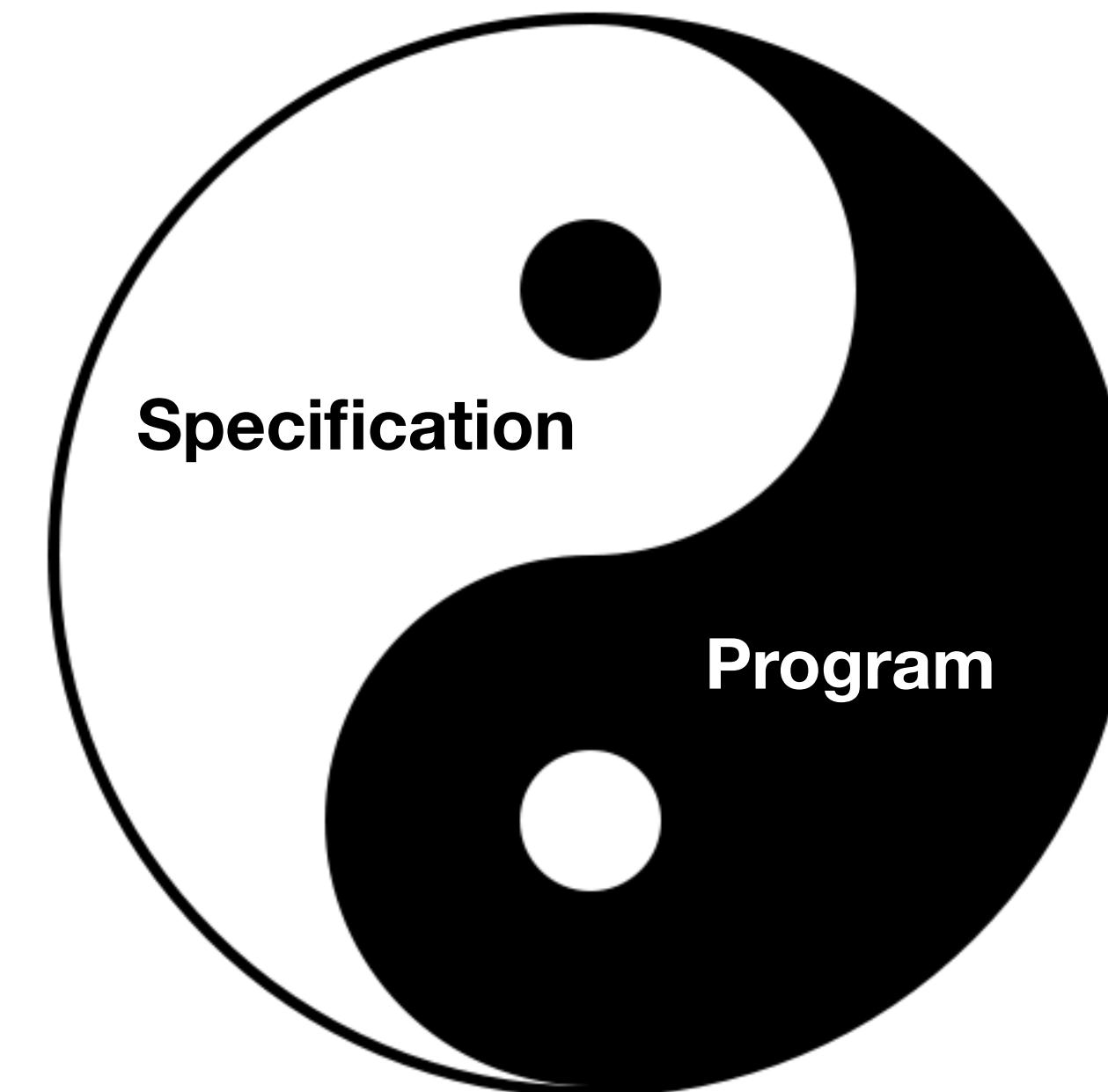
**20M+ New Developers
85M+ New Repositories
227M+ New Pull Requests
in 2022**

Software Complexity

less-382 (23,822 LOC)



What is the Future of Programming?



How to automatically **generate** a program
that satisfies the specification?

How to automatically **prove** if a given program
satisfies the specification?



Course Objectives: Principles

- Program verification
 - How to specify specification?
 - How to generate a correctness proof?
- Program synthesis
 - How to specify specification?
 - How to generate a correct program?

Success Stories

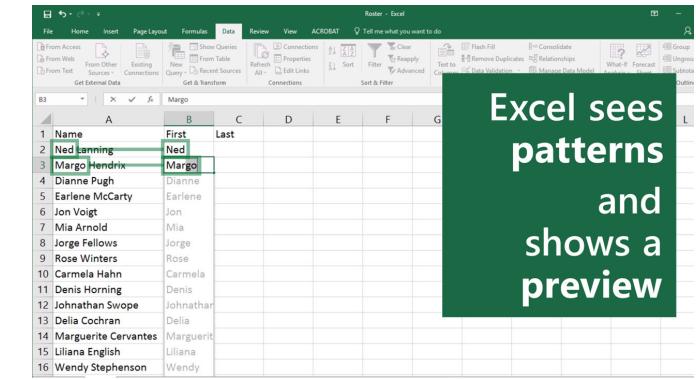
Program Verification



Windows Device Driver

Microsoft

Program Synthesis



Excel FlashFill

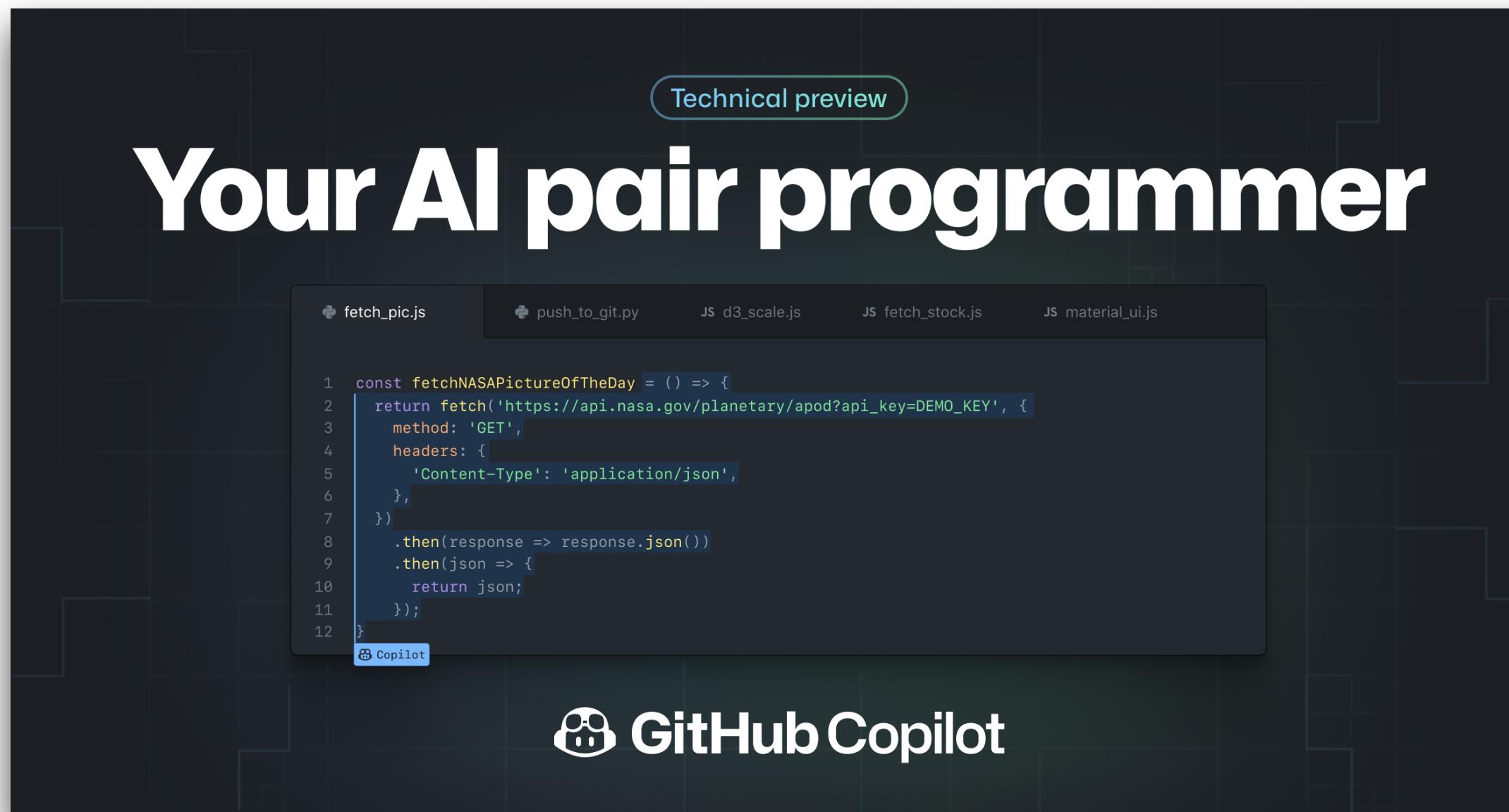
Microsoft

Astrée
Airbus Controller
ENS / AbsInt

A screenshot of a GitHub Copilot interface showing a file named "sentiments.ts". The code uses Node.js and the fetch API to determine if text is positive or negative by sending it to a sentiment analysis API. It includes imports for ts-node, fetch, and promises, and defines an isPositive function that sends a POST request to "https://text-processing.com/api/sentiment/" with the text as the body and JSON headers.

Copilot
Github

The Era of AI



A screenshot of the GitHub Copilot interface showing a Python script named "parse_expenses.py". The script defines a function "parse_expenses" that takes a string of expense lines and returns a list of triples (date, value, currency). It uses the datetime module to parse dates. The GitHub Copilot logo is visible at the bottom right of the code editor.

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7
8     Example expenses_string:
9         2016-01-02 -34.01 USD
10        2016-01-03 2.59 DKK
11        2016-01-03 -2.72 EUR
12
13     """
14     expenses = []
15     for line in expenses_string.splitlines():
16         if line.startswith("#"):
17             continue
18         date, value, currency = line.split(" ")
19         expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
20                           float(value),
21                           currency))
22
23     return expenses
```

Challenge: Program Correctness (1)

A screenshot of a code editor window titled 'int check_buffer_overflow() { Untitled-2'. The code contains a single function definition:

```
1 int check_buffer_overflow() {
2     int buffer[5];
3     buffer[6] = 0;
4     return 0;
5 }
```

```
1 // read an image file
2 int read_file() {
3     FILE *fp;
4     int i, j, k;
5     int n;
6     int m;
7     int nc;
8     int nr;
9     int nb;
10    int nc_max;
11    int nr_max;
12    int nb_max;
13    int nc_min;
14    int nr_min;
15    int nb_min;
16    int nc_avg;
17    int nr_avg;
18    int nb_avg;
19    int nc_sum;
20    int nr_sum;
21    int nb_sum;
22    int nc_var;
23    int nr_var;
24    int nb_var;
25    int nc_std;
26    int nr_std;
27    int nb_std;
28    int nc_med;
29    int nr_med;
30    int nb_med;
31    int nc_mode;
32    int nr_mode;
33    int nb_mode;
34    int nc_min_index;
35    int nr_min_index;
36    int nb_min_index;
37    int nc_max_index;
38    int nr_max_index;
39    int nb_max_index;
40    int nc_med_index;
41    int nr_med_index;
42    int nb_med_index;
43    int nc_mode_index;
44    int nr_mode_index;
45    int nb_mode_index;
46    int nc_sum_index;
47    int nr_sum_index;
48    int nb_sum_index;
49    int nc_var_index;
50    int nr_var_index;
51    int nb_var_index;
52    int nc_std_index;
53    int nr_std_index;
54    int nb_std_index;
55    int nc_avg_index;
56    int nr_avg_index;
57    int nb_avg_index;
58    int nc_min_index_index;
59    int nr_min_index
```

Challenge: Program Correctness (2)

- Recurring SW vulnerabilities by human developers
- Example: gimp-2.6.7 (CVE-2009-1570)

```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); } 3  
short ToS (char *pbuffer) { return ((short)(puffer[0] | puffer[1]<<8)); }  
gint32 ReadBMP (gchar *name, GError **error) { 1  
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)  
        FATALP ("BMP: Error reading BMP file header #3");  
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2  
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);  
  
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4  
    image_ID = ReadImage (rowbytes); 5  
    ...  
}  
  
gint32 ReadImage (int rowbytes) {  
    buffer = malloc(rowbytes); 6 // malloc with overflowed size  
    ...  
}
```

Challenge: Program Correctness (3)

- Recurring SW vulnerabilities by human developers
- Example: sam2p-0.49.4 (CVE-2017-1663)

```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); } 3  
short ToS (char *pbuffer) { return ((short)(puffer[0] | puffer[1]<<8)); }  
  
bitmap_type bmp_load_image (FILE* filename) { 1  
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)  
        FATALP ("BMP: Error reading BMP file header #3");  
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2  
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);  
  
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4  
    image.bitmap = ReadImage (rowbytes); 5  
    ...  
}  
  
unsigned char* ReadImage (int rowbytes) { 6  
    unsigned char *buffer = (unsigned char*) new char[rowbytes]; // malloc with overflowed size  
    ...  
}
```

Challenge: Program Correctness (4)

- Recurring SW vulnerabilities by AI developers
- Example (feat. Copilot):

```
int toLong(char *buffer) {
    return (buffer[0]) | (buffer[1] << 8) | (buffer[2] << 16) | (buffer[3] << 24);
}

int f(char *name) {
    int width, height, area;
    char buffer[10];
    FILE *fd = fopen(name, "rb");
    fread(buffer, 10, 1, fd);
    fclose(fd);

    // Copilot, fill it!
    width = toLong(buffer + 18);
    height = toLong(buffer + 22);
    area = width * height;
```

Course Objectives: Opportunities

- What is the state-of-the-art? What are the applications?
 - Program verification: API protocol, compiler, web browser, OS, DNN, ...
 - Program synthesis: cryptography, de-obfuscation, programming-by-example, ...
- What is your own application?

Course Objectives: Practice & Challenge

- Homework: **design & implement** program verifier & synthesizer
 - 6 (main) + 3 (dummy) assignments
- Programming assignments in OCaml using LLVM & Z3
 - You will write your verifier & synthesizer in OCaml
 - Your verifier will analyze LLVM IR code
 - You will utilize Z3 in your verifier & synthesizer
- Why LLVM? (<https://llvm.org>)
- Why OCaml? (<https://ocaml.org>)
- Why Z3? (<https://github.com/Z3Prover/z3>)



The LLVM Compiler Infrastructure

- The de-facto standard & well-structured compiler toolchain
 - parser, code optimizer, linker, loader, debugger, etc
- A wide variety of frontends: C/C++, Obj-C, Swift, Fortran, etc
 - translated to the LLVM IR (intermediate representation)



Apple's other open secret: the LLVM Compiler

By Prince McLean

Friday, June 20, 2008, 04:10 am PT (07:10 am ET)

SproutCore, profiled earlier this week, isn't the only big news spill out from the top secret WWDC conference due to Apple's embrace of open source sharing. Another future technology featured by the Mac maker last week was LLVM, the Low Level Virtual Machine compiler infrastructure project.

Like SproutCore, LLVM is neither new nor secret, but both have been hiding from attention due to a thick layer of complexity that has obscured their future potential.

Looking for LLVM at WWDC

Again, the trail of breadcrumbs for LLVM starts in the public WWDC schedule. On Tuesday, the session "New Compiler Technology and Future Directions" detailed the following synopsis:

Google Chrome is replacing Microsoft's C++ compiler with Clang

By Muhammad Jarir Kanji · Mar 6, 2018 14:06 EST · HOT!



Alongside bringing better touch support and automatic ad-blocking for 'intrusive' ads to the desktop version of Chrome, Google is also making some changes to its browser under the hood. The company is now starting to build Chrome for Windows using the Clang compiler which it already uses for other platforms like macOS and Linux.

IBM Developer

Power developer portal Blogs Feedback

Announcements Compilers

IBM C/C++ and Fortran compilers to adopt LLVM open source infrastructure

SiyuanZhang

Published on February 23, 2020 / Updated on February 26, 2020

The OCaml Language

- Simple, safe, and realistic programming language
 - Strong type system, higher-order functions, etc
- Official OCaml bindings to LLVM and Z3 API supported
- A lot of growing demands from academia and industry
- See the materials on the course webpage



The Z3 Theorem Prover



- State-of-the-art automated theorem prover by Microsoft Research
- Solving satisfiability modulo theory (SMT) problems
 - first-order logic with background theories
(e.g., arithmetic, bit-vectors, arrays, datatypes, uninterpreted functions, etc)

Boolean Satisfiability Problem (SAT)

$$(\neg A \vee B) \wedge (\neg B \vee C) \wedge (A \vee \neg C \vee B)$$

Satisfiable when

$A = \text{false}$

$B = \text{true}$

$C = \text{true}$

Satisfiability modulo theory (SMT)

$$x + 2 = y \implies f \text{read } write(a, x, 3, y - 2) = f(y - x + 1)$$

Arithmetic

Array

Uninterpreted Functions

Homework

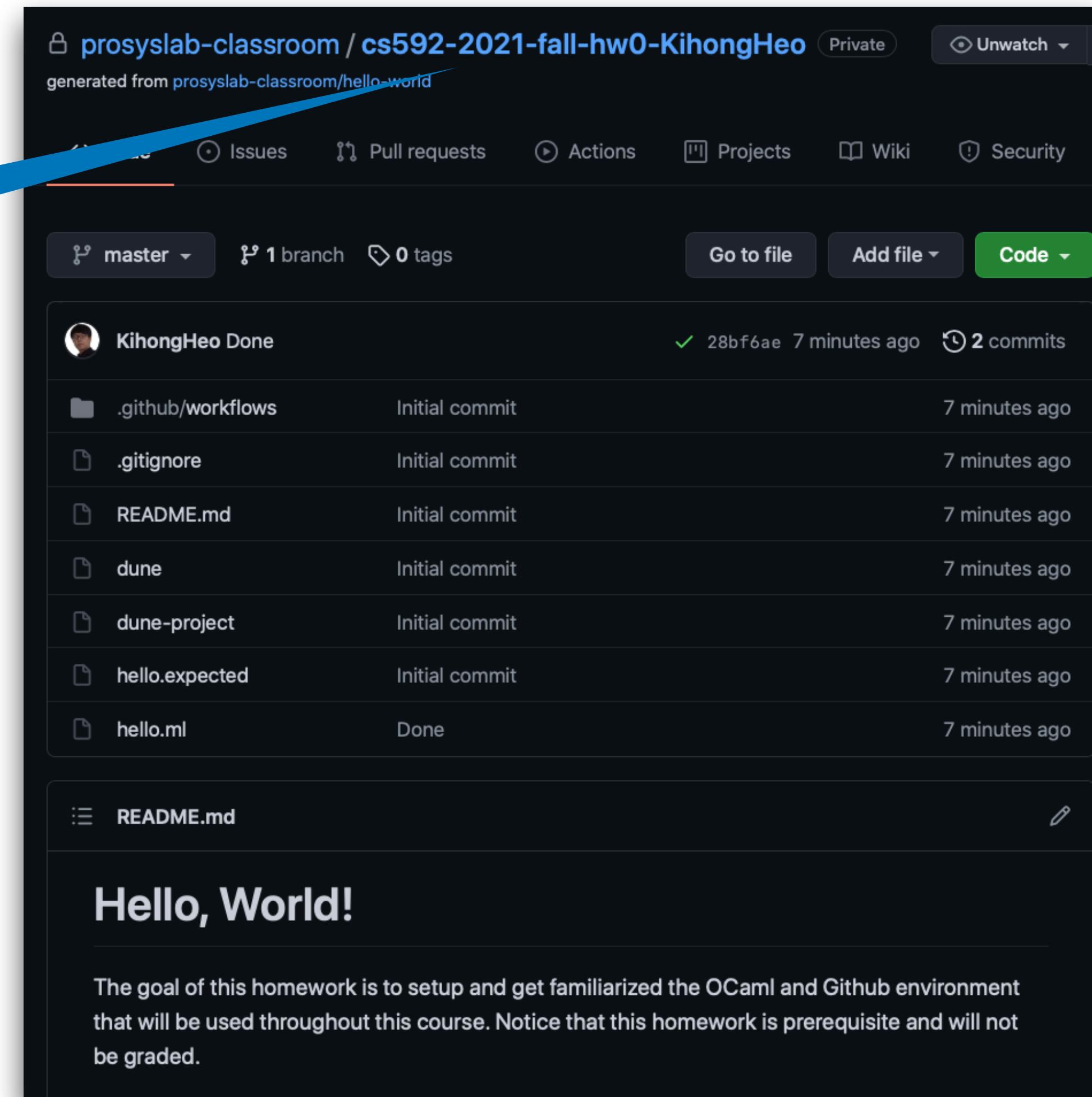
- All submissions will be managed using Github / Github Classroom
 1. For each HW, a unique invitation URL will be posted on the issue board
 2. Once you accept, a private repo for your assignment will be created
 3. You can push as many commits as you want before the deadline
 4. The final commit of your master branch will be graded
- Gradescope will be used for some assignments and exams
- 80% credit for 1-day late, 50% credit for 2-days late, NO credit otherwise

Homework 0.1: Hello, World!

- Goal: setting up and getting familiarized with OCaml and Git environments
 - Implement your “hello-world” program in OCaml
 - Test on your machine
 - Push to your Github repository
 - See the result in Github Action
- The invitation URL will be posted on the course webpage
- Will not be graded

Homework 0.1: Hello, World!

1. Accept the invitation
and have your repository



Homework 0.1: Hello, World!

A screenshot of a GitHub repository page for "prosylab-classroom / cs592-2021-fall-hw0-KihongHeo". The repository is private and was generated from "prosylab-classroom/hello-world". The "Code" tab is selected, showing the master branch with 1 branch and 0 tags. A commit by "KihongHeo Done" was made 7 minutes ago, containing 2 commits. The commit details show files like .github/workflows, .gitignore, README.md, dune, dune-project, hello.expected, and hello.ml. The README.md file contains the text "Hello, World!". A blue callout box on the left points to the commit history with the text "2. Commit your code".

2. Commit your code

prosylab-classroom / cs592-2021-fall-hw0-KihongHeo Private

generated from prosylab-classroom/hello-world

Code Issues Pull requests Actions Projects Wiki Security

master 1 branch 0 tags Go to file Add file Code

KihongHeo Done 28bf6ae 7 minutes ago 2 commits

| File | Type | Description | Time |
|-------------------|----------------|---------------|------|
| .github/workflows | Initial commit | 7 minutes ago | |
| .gitignore | Initial commit | 7 minutes ago | |
| README.md | Initial commit | 7 minutes ago | |
| dune | Initial commit | 7 minutes ago | |
| dune-project | Initial commit | 7 minutes ago | |
| hello.expected | Initial commit | 7 minutes ago | |
| hello.ml | Done | 7 minutes ago | |

README.md

Hello, World!

The goal of this homework is to setup and get familiarized the OCaml and Github environment that will be used throughout this course. Notice that this homework is prerequisite and will not be graded.

Homework 0.1: Hello, World!

3. See your result

The screenshot shows a GitHub repository page for `prosylab-classroom / cs592-2021-fall-hw0-KihongHeo`. The repository is private and was generated from `prosylab-classroom/hello-world`. The 'Code' tab is selected, showing the master branch with 1 branch and 0 tags. There are 2 commits by KihongHeo Done, both made 7 minutes ago. The commits are:

| File | Type | Message | Time |
|--------------------------------|----------------|---------|---------------|
| <code>.github/workflows</code> | Initial commit | 28bf6ae | 7 minutes ago |
| <code>.gitignore</code> | Initial commit | 28bf6ae | 7 minutes ago |
| <code>README.md</code> | Initial commit | 28bf6ae | 7 minutes ago |
| <code>dune</code> | Initial commit | 28bf6ae | 7 minutes ago |
| <code>dune-project</code> | Initial commit | 28bf6ae | 7 minutes ago |
| <code>hello.expected</code> | Initial commit | 28bf6ae | 7 minutes ago |
| <code>hello.ml</code> | Done | 28bf6ae | 7 minutes ago |

The `README.md` file content is:

```
Hello, World!
```

The goal of this homework is to setup and get familiarized the OCaml and Github environment that will be used throughout this course. Notice that this homework is prerequisite and will not be graded.

Homework 0.2: OCaml Programming

- Goal: getting familiarized with basic OCaml programming
 - Solve a few programming problems in OCaml
 - Test on your machine
 - Push to your Github repository
 - See the result in Github Action
- The invitation URL will be posted on the course webpage
- Will not be graded but highly recommended if you are not familiar with OCaml

Requirement: Software Engineering Practices

- All programming assignments submissions must follow basic SE practices

- Remove all compile errors and warning

- Remember: warning is error

- Clean code via formatting

- Acceptable code coverage

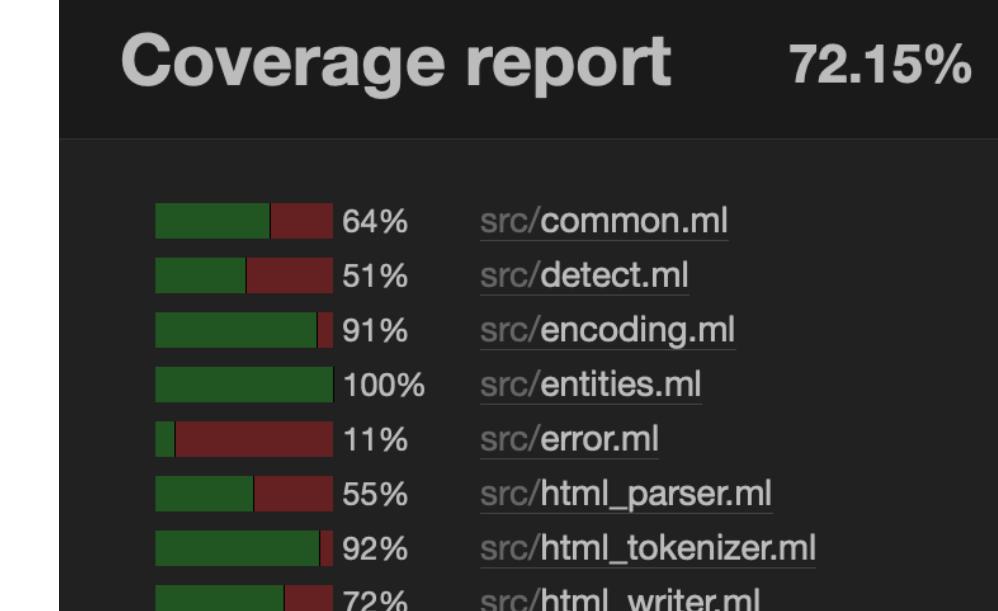
- Otherwise, NO points will be given

- Detailed instructions will be provided

```
File "src/semantics.ml", line 39, characters 6-11:  
39 |   let count = 1 in  
      ^^^^  
Warning 26 [unused-var]: unused variable count.
```

```
let rec  
  fact n =  
    match n with  
    | 0 | 1 -> 1  
    | _ -> n *  
        fact (n - 1)
```

```
let rec fact n =  
  match n with  
  | 0 | 1 -> 1  
  | _ -> n * fact (n - 1)
```



```
14 module Ns :  
15 sig  
16   val to_string : ns -> string  
17 end =  
18 struct  
19   let to_string = function  
20   | HTML -> html_ns  
21   | MathML -> mathml_ns  
22   | SVG -> svg_ns  
23   | Other s -> s  
24 end
```

Homework 0.3: Watching the Video

- Watch the Youtube video about “Undecidability” vis the course webpage
- Think about the followings:
 - What is the fundamental limitation of SW?
 - Why is it difficult to prove the correctness of SW?
 - How can we overcome the above challenges?
- Provide your opinion and write an essay
 - Not mandatory but a reward might be given depending on the quality
 - E.g., participation point, questions for the final exam, drink, fame, etc

Writing

- Syntactic requirements
 - Typeset your document in Latex using a provided template (maximum 2 pages)
 - Korean (if you are a native Korean speaker), English (otherwise)
- Semantic requirements: Top-down (두괄식)
 - For each paragraph, write the topic sentence first followed by the details.
- See the README for the details
- The invitation URL will be posted on the course webpage

Misc

- Submit your Github account via the google form (see the Github issue board)
- Join Gradescope (see the Github issue board)
 - For written assignments and the final exam
- Rules for programming assignments
 - Preserve the structures (directories, files, types, etc)
 - Don't install further Github App