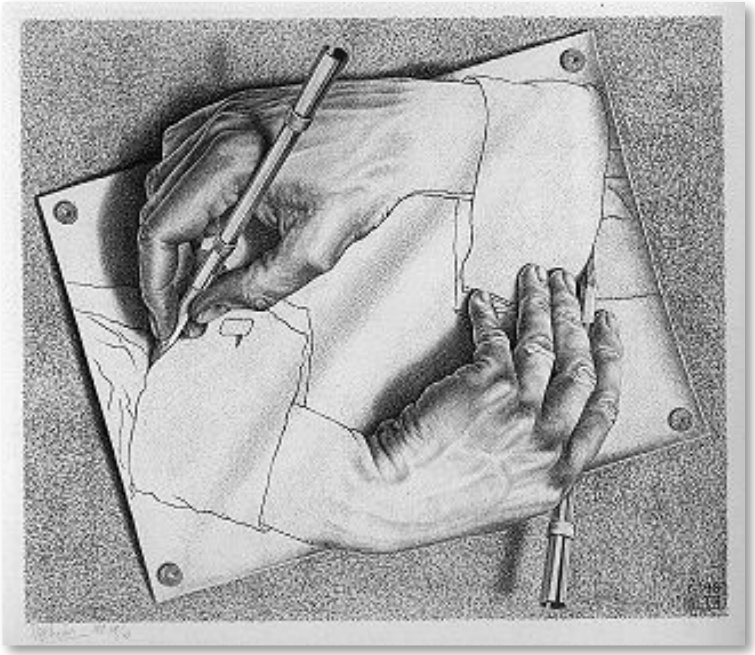
Programming Languages and Techniques (CIS120)

Bonus Lecture

April 25, 2016

“Code is Data”

Code is Data



M.C. Escher, Drawing Hands, 1948

Code *is* Data

- A Java source file is just a sequence of characters.
- We can represent programs with Strings!

```
String p_0 = "class C { public static void main(String args[])
```

```
String p_1 = "class C { public static void main(String args[])
```

```
String p_2 = "class C { public static void main(String args[])
```

```
String p_3 = "class C { public static void main(String args[])
 {System.out.println(\"Hel\");}}";
```

```
String p_13 = "class C { public static void main(String args[])
 {System.out.println(\"Hello, world!\");}}";
```

…

```
String p_12012023423123O = /* Mushroom of Doom! */
      "class Game { public static void main(String args[]) {…}}";
```

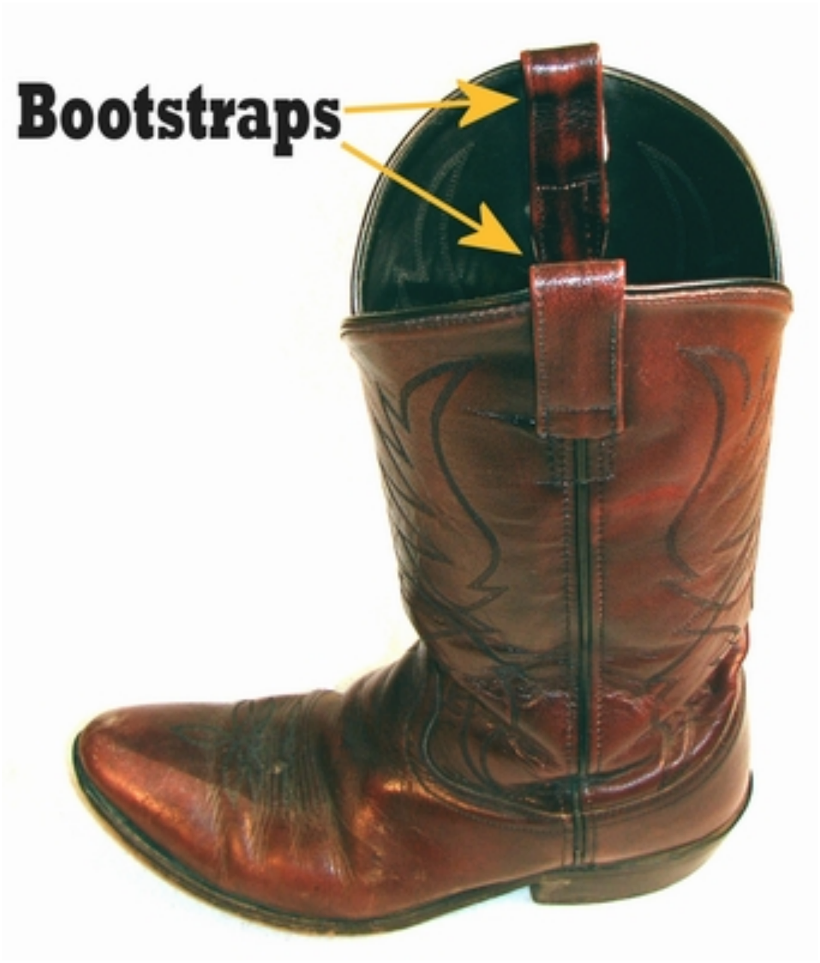
…

```
String p_99993249000902300239400823407O234 = /* Minecraft! */
      "class Minecraft { public static void main(String args[]) {…}}";
```

…

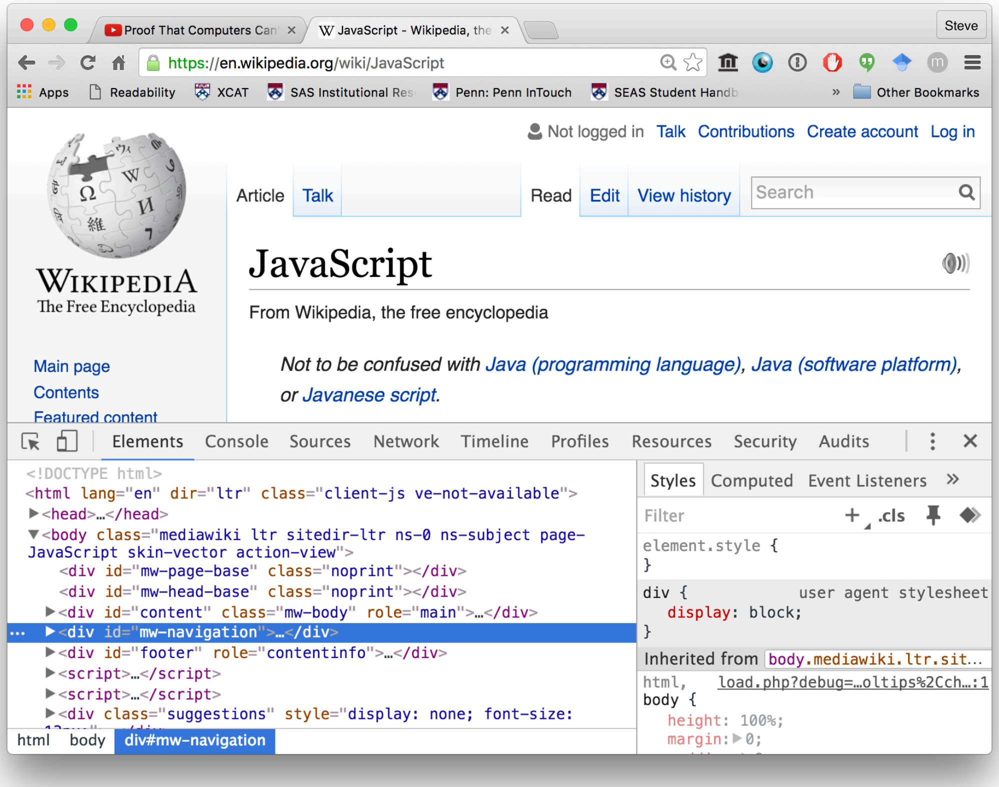
```
String p_99234992342399999324900023428234073450234534 = /* Eclipse! */
      "class Eclipse { public static void main(String args[]) {…}}";
```

Consequence 1: Programs that manipulate programs



Interpreters

- We can create *programs* that manipulate *programs*
- An *interpreter* is a program that executes other programs
- `interpret ("3 + 4")` ➔ 7
- Example 1: JavaScript



JavaScript

From Wikipedia, the free encyclopedia

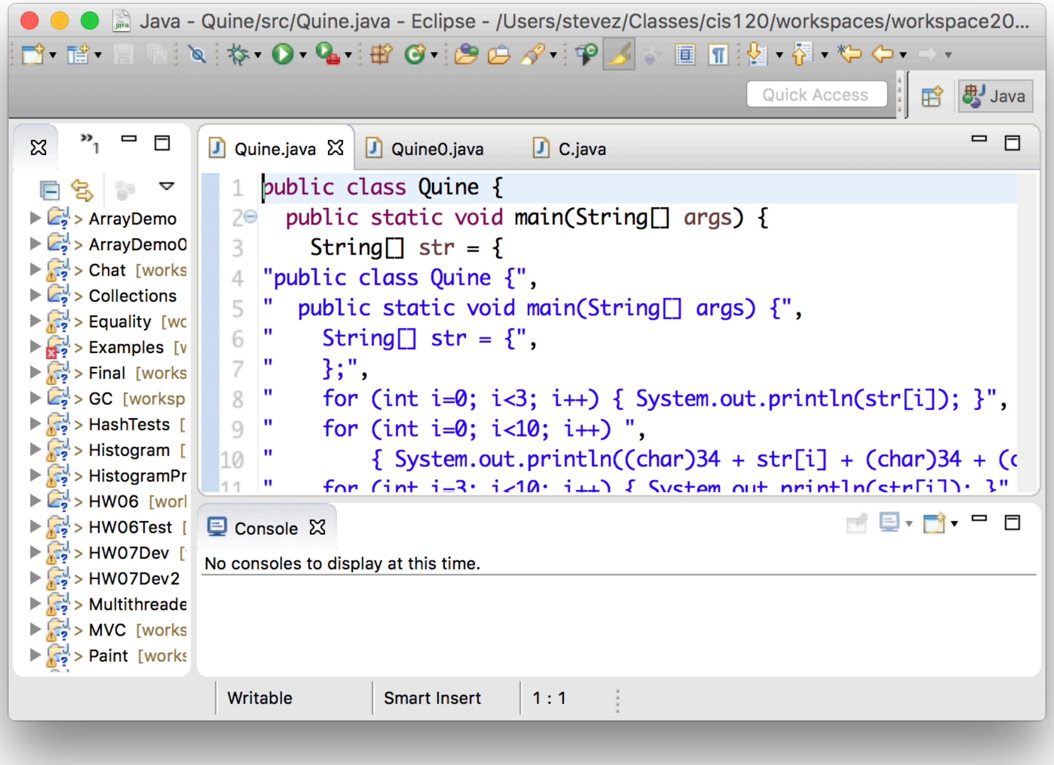
Not to be confused with Java (programming language), Java (software platform), or Javanese script.

```
<!DOCTYPE html>
<html lang="en" dir="ltr" class="client-js ve-not-available">
▶<head>…</head>
▼<body class="mediawiki ltr sitedir-ltr ns-0 ns-subject page-
JavaScript skin-vector action-view">
    <div id="mw-page-base" class="noprint"></div>
    <div id="mw-head-base" class="noprint"></div>
  ▶<div id="content" class="mw-body" role="main">…</div>
… ▶<div id="mw-navigation">…</div>
  ▶<div id="footer" role="contentinfo">…</div>
  ▶<script>…</script>
  ▶<script>…</script>
  ▶<div class="suggestions" style="display: none; font-size:
```

```
element.style {
}
div {                    user agent stylesheet
   display: block;
}
Inherited from body.mediawiki.ltr.sit…
html,    load.php?debug=…oltips%2Cch…:1
body {
   height: 100%;
   margin:▶0;
```

IDEs and Compilers

- Example 2: Eclipse
  - Note that Eclipse manipulates a *representation* of Java programs
  - Eclipse itself is written in Java
  - So you could use Eclipse to edit the code for Eclipse… ?!



```
public class Quine {
  public static void main(String[] args) {
    String[] str = {
"public class Quine {",
"  public static void main(String[] args) {",
"    String[] str = {",
"    };",
"    for (int i=0; i<3; i++) { System.out.println(str[i]); }",
"    for (int i=0; i<10; i++) ",
"        { System.out.println((char)34 + str[i] + (char)34 + (c
"    for (int i=3; i<10; i++) { System.out.println(str[i]); }"
```

```
No consoles to display at this time.
```

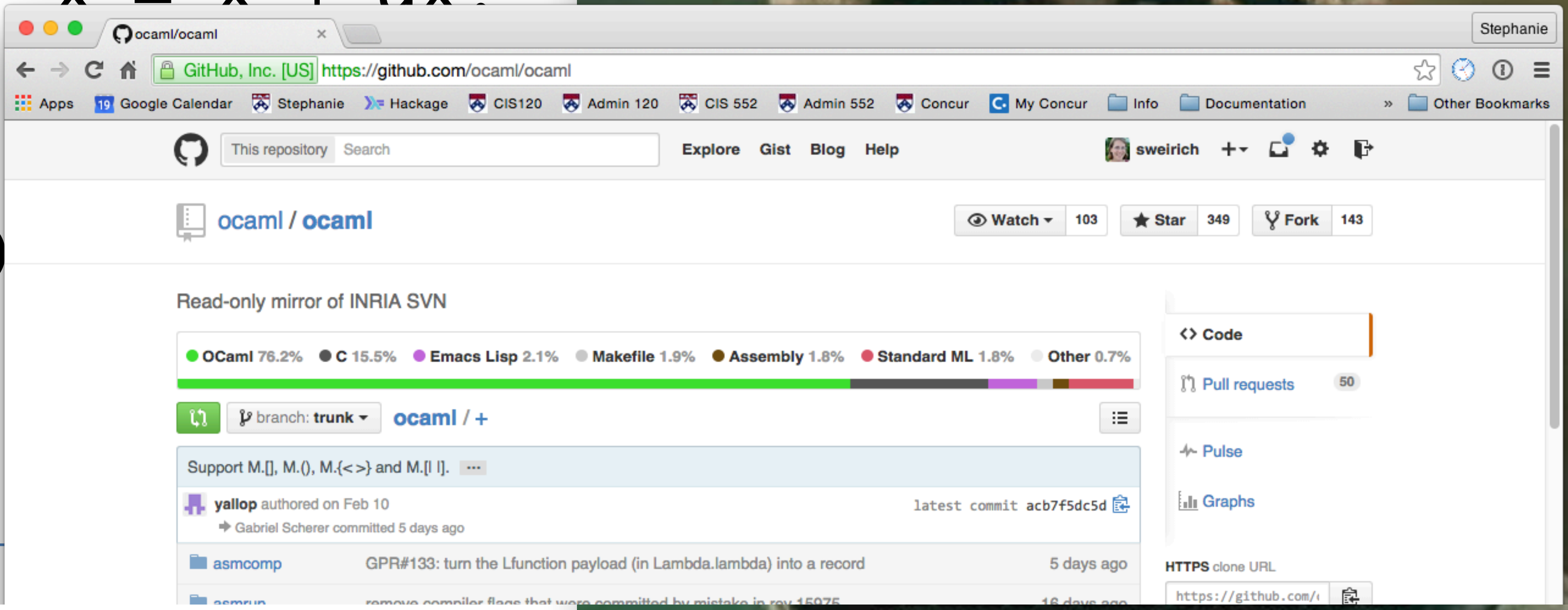
- Example 3: Compiler
  - The Java compiler takes a representation of a Java program
  - It outputs a “low-level” representation of the program as a .class file (i.e. Java byte code)
  - Can also compile to other representations, e.g. x86 “machine code”

Example Compilation: Java to X86

```
class Point {
   int x;
   int y;
   Point move(int
  int dy) {
     x = x + dx;

   )
}
```

```
.globl __fun__Point.move
__fun__Point.move:
      pushl %ebp
      movl %esp, %ebp
      subl $4, %esp
__5:
      movl 8(%ebp), %eax
      movl 4(%eax), %eax
      movl %eax, -4(%ebp)
```



ocaml / ocaml

Read-only mirror of INRIA SVN

OCaml 76.2% C 15.5% Emacs Lisp 2.1% Makefile 1.9% Assembly 1.8% Standard ML 1.8% Other 0.7%

ocaml / +

Support M.[], M.(), M.{<>} and M.[| |]. ...

yallop authored on Feb 10 latest commit acb7f5dc5d
Gabriel Scherer committed 5 days ago

asmcomp	GPR#133: turn the Lfunction payload (in Lambda.lambda) into a record	5 days ago
asmrun	remove compiler flags that were committed by mistake in rev 15075	16 days ago

Consequence 2: Malware

Rene Magritte, The Human Condition, 1933

Consequence 2: Malware

- Why does Java do array bounds checking?
- *Unsafe* language like C and C++ don't do that checking;
  - They will happily let you write a program that "writes past" the end of an array.

How to make money onli... — spamfolder

Keri.Day@seas.upenn.e

To: Stephanie Weirich

How to make money onli

March 1, 2016 at 4:59 AM

- Result:
  - viruses, worms, "jailbreaking" mobile phones, Spam, botnets, …
- Fundamental issue:
  - Code is data.
  - Why?

Consider this C Program

```
void m() {
  char[2] buffer;

  char c = read();
  int i = 0;
  while (c != -1) {
    buffer[i] = c;
    c = read();
    i++;
  }
  process(buffer);
}

void main() {
  m();
  // do some more stuff
}
```

Notes:

- C doesn’t check array bounds
- Unlike Java, it stores arrays directly on the stack
- What could possibly go wrong?

Abstract Stack Machine

“Stack Smashing Attack”

Abstract Stack Machine

Workspace

Stack

```
m();
// do some more stuff
```

Call to main() to start the program…

Abstract Stack Machine

Workspace

```
char[2] buffer;

char c = read();
int i = 0;
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```

Push the saved workspace, run m()

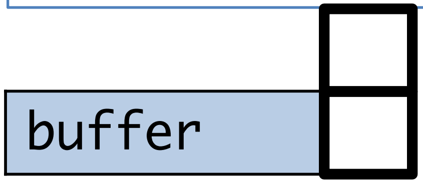
Abstract Stack Machine

Workspace

```
char c = read();
int i = 0;
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



Allocate space for buffer on the stack.

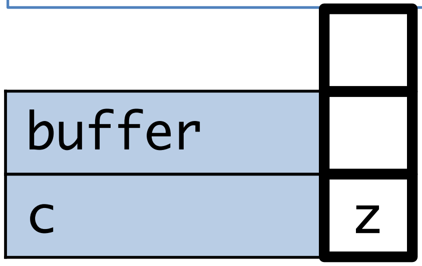
Abstract Stack Machine

Workspace

```
int i = 0;
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



buffer

c

z

Allocate space for c.
Read the first user input... 'z'.

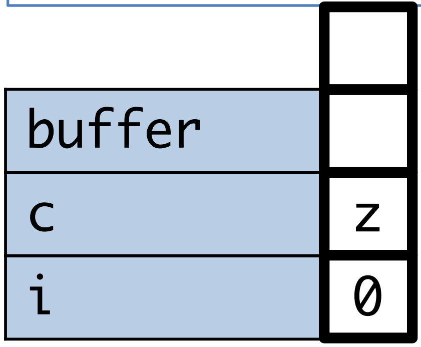
Abstract Stack Machine

Workspace

```
while (c != -1) {
   buffer[i] = c;
   c = read();
   i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



buffer	
c	z
i	0

Allocate space for i.

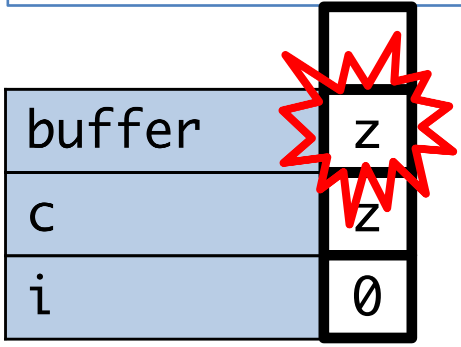
Abstract Stack Machine

Workspace

```
while (c != -1) {
  buffer[i]  = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



Copy (contents of) c to buffer[0]

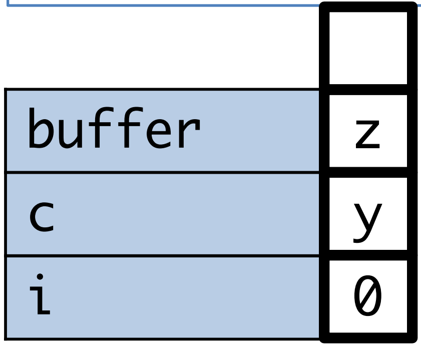
Abstract Stack Machine

Workspace

```
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



buffer	z
c	y
i	0

Read next character … 'y'

Abstract Stack Machine

Workspace

```
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```

buffer	z
c	y
i	1

Increment i

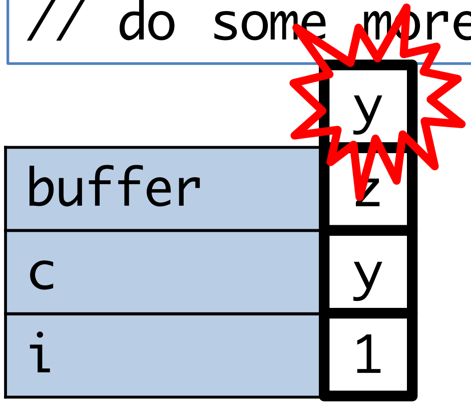
Abstract Stack Machine

Workspace

```
while (c != -1) {
   buffer[i] = c;
   c = read();
   i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



	y
buffer	z
c	y
i	1

Copy (contents of) c to buffer[1]

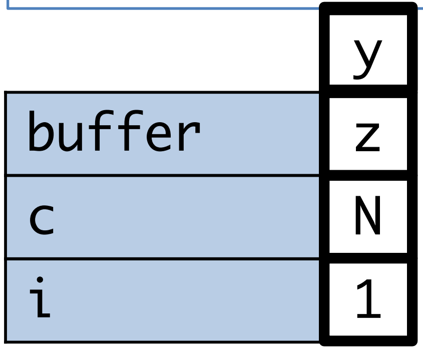
Abstract Stack Machine

Workspace

```
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



Read next character … 'N'

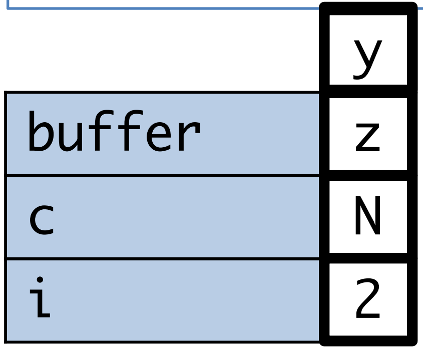
Abstract Stack Machine

Workspace

```
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

```
_;
// do some more stuff
```



	y
buffer	z
c	N
i	2

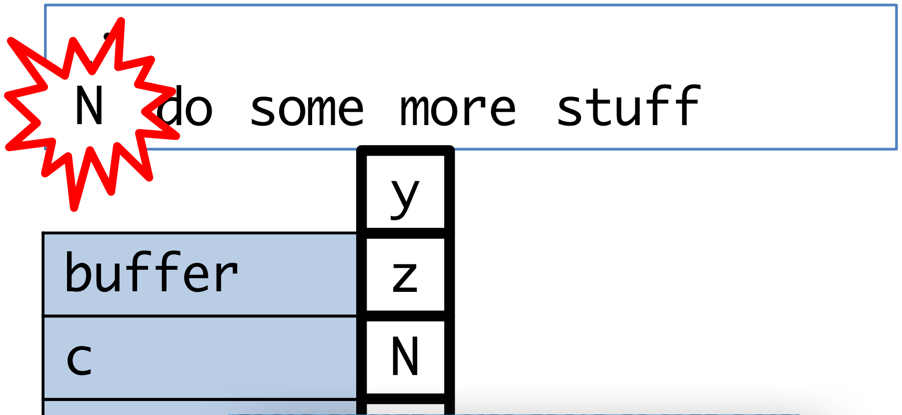
Increment i

Abstract Stack Machine

Workspace

```
while (c != -1) {
   buffer[i] = c;
   c = read();
   i++;
}
process(buffer);
```

Stack



N do some more stuff

	y
buffer	z
c	N
i	

Copy (contents of) c to buffer[2] ?!?

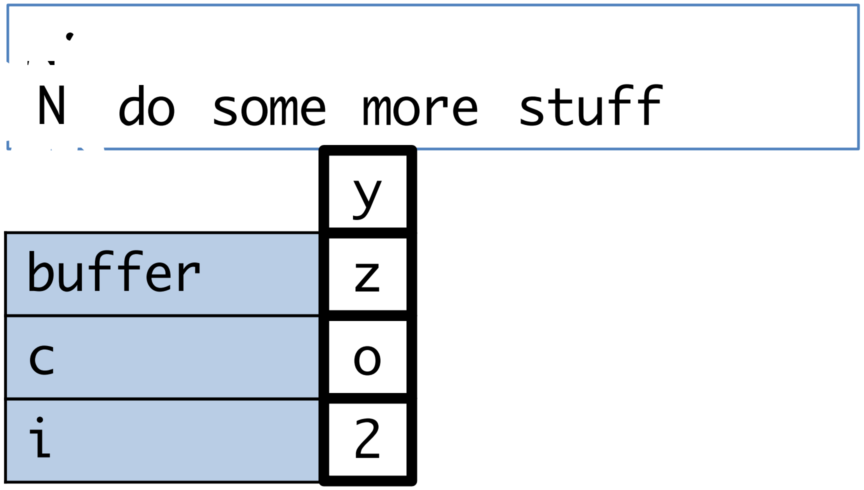
Overwrites the saved workspace!?

Abstract Stack Machine

Workspace

```
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack



N do some more stuff

	y
buffer	z
c	o
i	2

Keep going… read 'o'…

Abstract Stack Machine

Workspace

```
while (c != -1) {
  buffer[i] = c;
  c = read();
  i++;
}
process(buffer);
```

Stack

N do some more stuff

	y
buffer	z
c	o
i	3

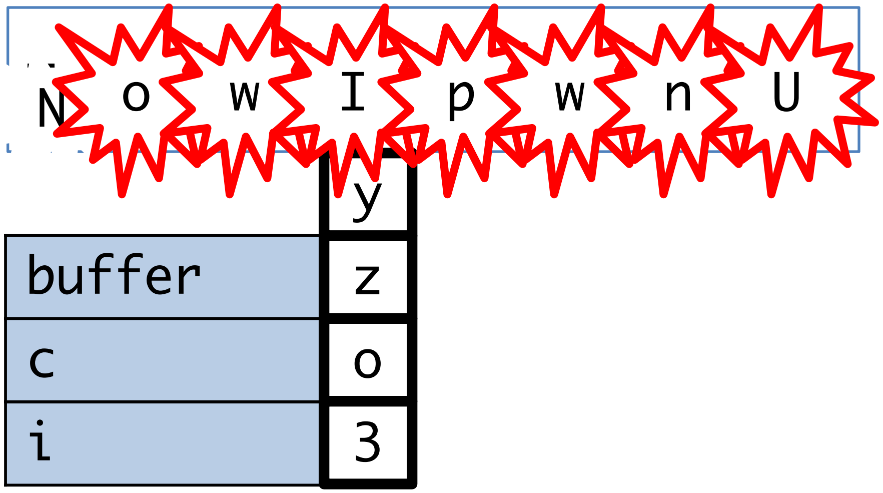
Keep going... read 'o'...increment i...

Abstract Stack Machine

Workspace

```
while (c != -1) {
   buffer[i] = c;
   c = read();
   i++;
}
process(buffer);
```

Stack



buffer	z
c	o
i	3

Keep going... read 'o'...increment i...write 'o' into saved workspace...

Abstract Stack Machine

Workspace

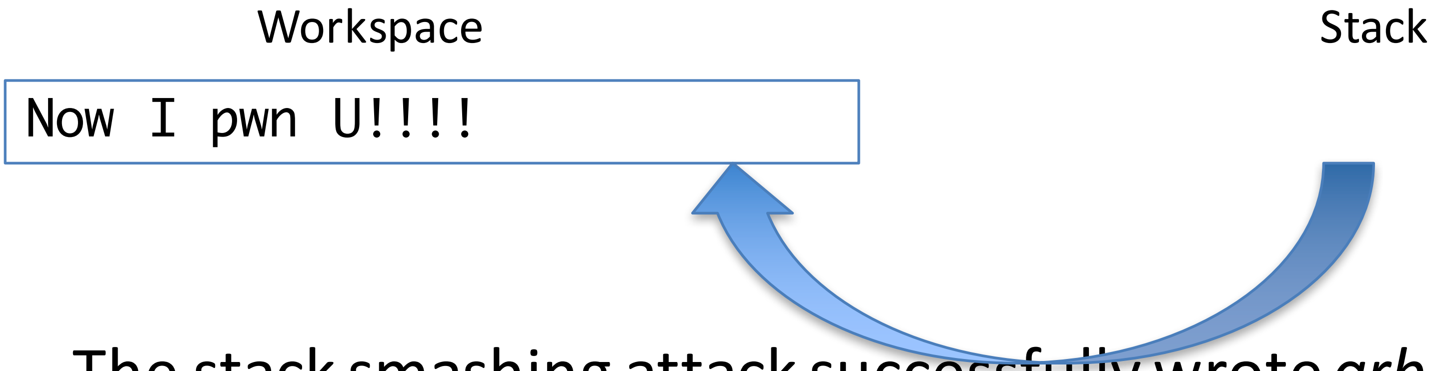
Stack

```
Now I pwn U!!!!
```

buffer	z
c	o
i	3

Later…

Abstract Stack Machine



Workspace

Stack

```
Now I pwn U!!!!
```

The stack smashing attack successfully wrote *arbitrary* code into the program's workspace...

RogueWave SOFTWARE

CodeBuzz

The Top Five Cyber Security Vulnerabilities

POSTED IN GENERAL SECURITY, INCIDENT RESPONSE ON JULY 2, 2015

US-CERT

UNITED STATES COMPUTER EMERGENCY READINESS TEAM

Buffer overflows are the top software security vulnerability of the past 25 years

ON MAR 11, 13 • BY CHRIS BUBINAS • WITH 2 COMMENTS

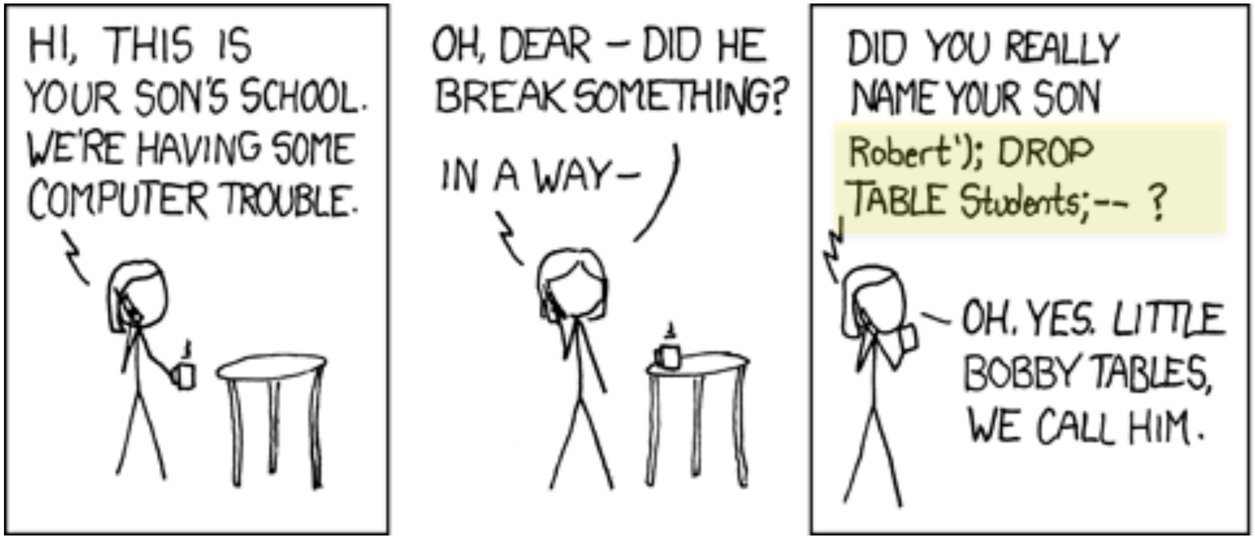
In a report analyzing the entire CVE and NVD databases, which date back to 1988, Sourcefire

Other Code Injection Attacks

```
void registerStudent() {
  print("Welcome to student registration.");
  print("Please enter your name:");
  String name = readLine();
  evalSQL("INSERT INTO Students('" + name + "')"  );
}
```

```
"INSERT INTO Students('Robert'); DROP TABLE Students; --')"
```

```
          + "Robert'); DROP TABLE Students; --" + "')"
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.

http://xkcd.com/327/

Consequence 3: Undecidability

Undecidability Theorem

Theorem: *It is impossible to write a method*

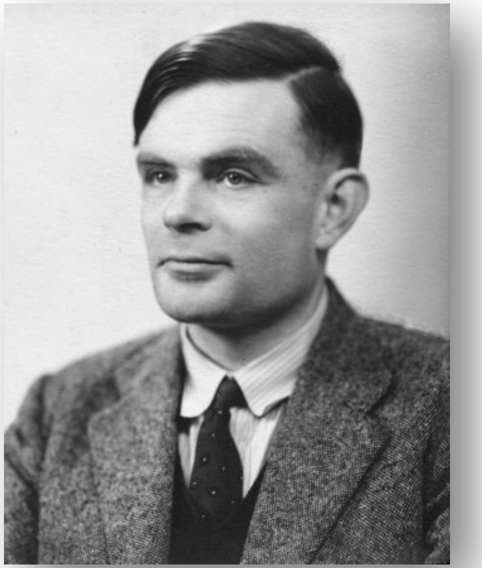
```
      boolean halts(String prog)
```

such that, for any valid Java program P represented as a string `p_P`,

```
      halts(p_P)
```

returns true exactly when the program P halts, and false otherwise.

Alonzo Church, April 1936



Alan Turing, May 1936

Halt Detector

- Suppose we could write such a program:

```
class HaltDetector {
  public static boolean halts(String javaProgram) {
    // ...do some super-clever analysis...
    // return true if javaProgram halts
    // return false if javaProgram does not
  }
}
```

- A correct implementation of HaltDetector.halts(p) always returns either true or false
  - i.e., it never raises an exception or loops
- HaltDetector.halts(p) ⇒ true means “p halts”
- HaltDetector.halts(p) ⇒ false means “p loops forever”

Do these methods halt?

```
“boolean m(){ return false; }”
```

⇒ YES

```
“boolean m(){ return m(); }”
```

⇒ NO (assuming infinite stack space)

```
“boolean m(){
   if (“abc”.length() == 3) return true;
   else return m(); }”
```

⇒ YES

```
“boolean m(){
   String x = “”;
   while (true) {
      if (x.length() == 3) return true;
      x = x + \“a\”;
    }
  return false;
}”
```

⇒ YES

Do these methods halt?

```
boolean m(){ return false; }
```

⇒ YES

```
boolean m(){ return m(); }
```

⇒ NO (assuming infinite stack space)

```
boolean m() {
    if ("abc".length() == 3 ) return true;
    else return m();
}
```

⇒ YES

Does this method halt for *all* n?

```
boolean m(int n) {
   if (n<=1) return true;
   else if ((n%2) == 0) return m(n/2);
   else return m(3*n + 1);
}
```

Assuming infinite amount of stack space and arbitrarily large integers, it is *unknown* whether this program halts for all (strictly) positive integers!

Collatz Conjecture proposed in 1937!

Consider this Program called Q:

```java
class HaltDetector {
  public static boolean halts(String javaProgram) {
    // ...do some super-clever analysis...
    // return true if javaProgram halts
    // return false if javaProgram does not
  }
}

class Main {
  public static void Q() {
     String p_Q = ???;   // string representing Q
     if (HaltDetector.halts(p_Q)) {
       while (true) {}  // infinite loop!
     }
  }
}
```

What happens when we run Q?

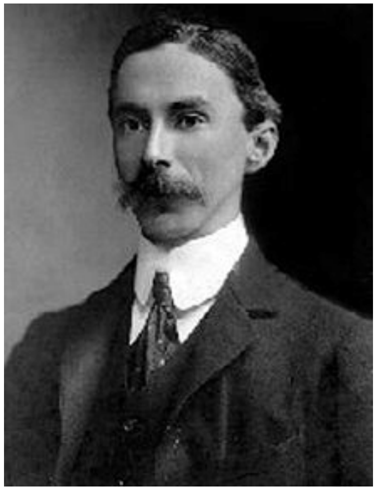
```
public static void Q() {
    String p_Q = ???;    // string representing Q
    if (HaltDetector.halts(p_Q)) {
      while (true) {}  // infinite loop!
    }
  }
```

if HaltDetector.halts(p_Q) ⇒ true then Q ⇒ infinite loop

if HaltDetector.halts(p_Q) ⇒ false then Q ⇒ halts

Contradiction!

- Russell's Paradox (1901)
- Gödel's Incompleteness Theorem (1931)
- Both rely on *self reference*



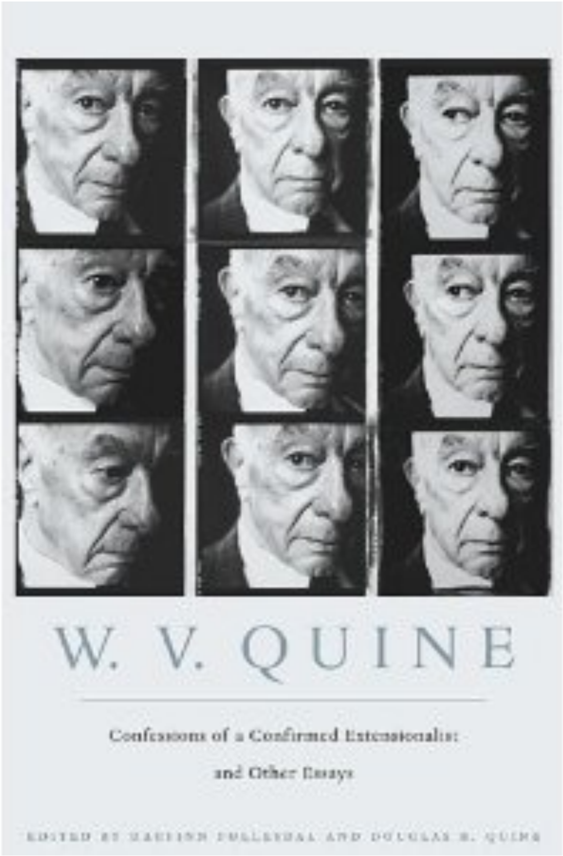
Bertrand Russell, 1901

Kurt Gödel, 1931

Potential Hole in the Proof

- What about the ??? in the program Q?
- It is supposed to be a String representing the program Q itself.
- How can that be possible?
- Answer: code is data!
  - And there's more than one representation for the same data.
- See Quine.java

YO, I'M M.C. QUINE AND I'M HERE TO SAY, "YO, I'M M.C. QUINE AND I'M HERE TO SAY!"


W. V. QUINE
Confessions of a Confirmed Extensionalist
and Other Essays
EDITED BY DAGFINN FØLLESDAL AND DOUGLAS B. QUINE

Profound Consequences

- The "halting problem" is *undecidable*
  - *There are problems that cannot be solved by a computer program!*

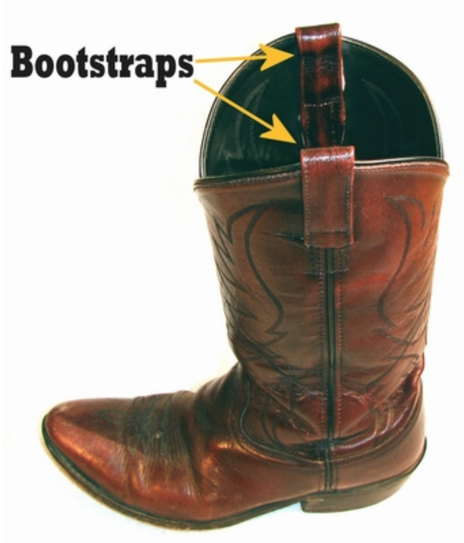
- Rice's Theorem:
  - Every "interesting" property about computer programs is undecidable!

- You can't write a perfect virus detector!
  (whether a program is a virus is certainly interesting)
  1. virus detector might go into an infinite loop
  2. it gives you false positives (i.e. says something is a virus when it isn't)
  3. it gives you false negatives (i.e. it says a program is not a virus when it is)

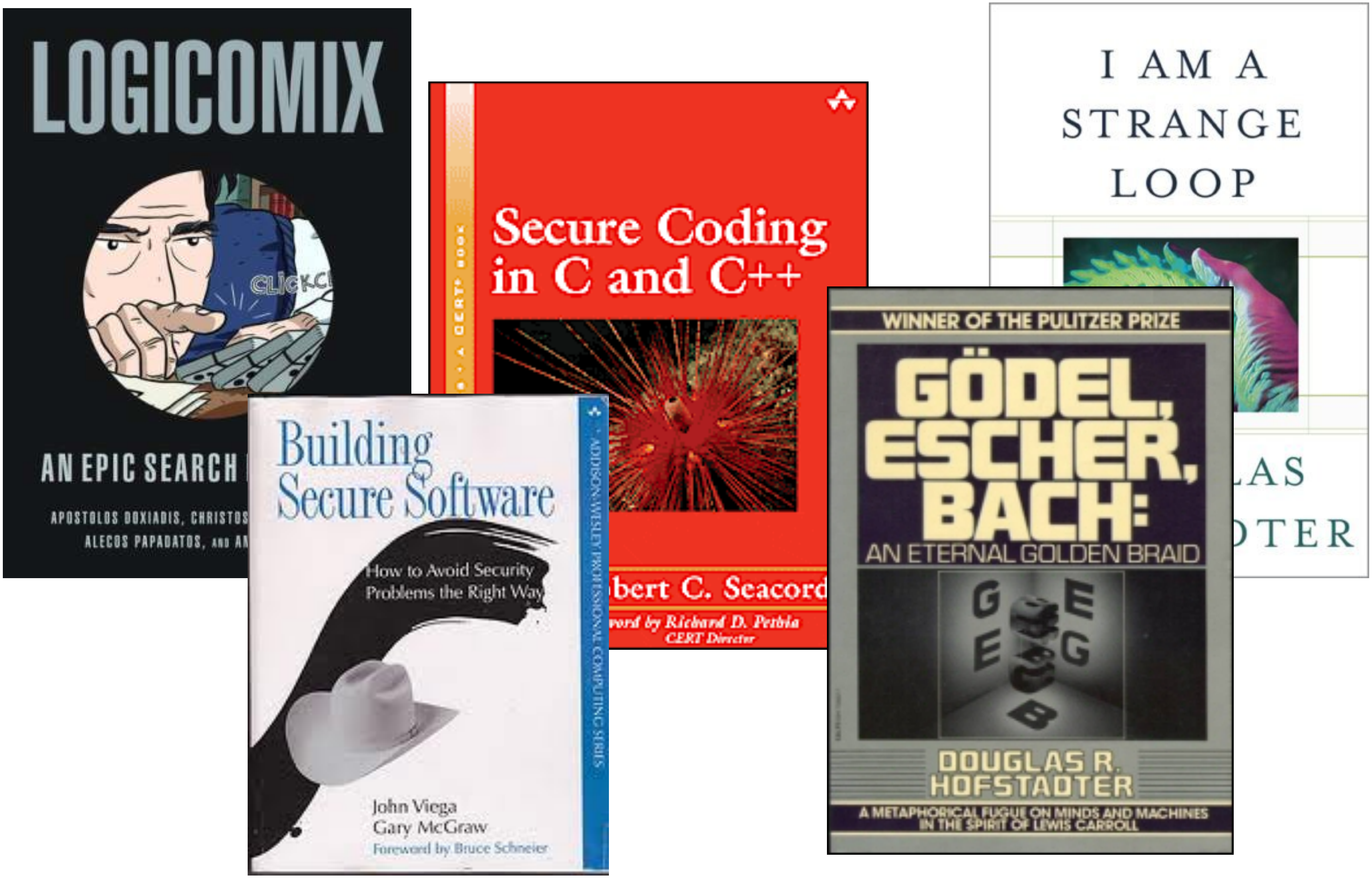
- Also: You can't write a perfect autograder!
  (whether a program is correct is certainly interesting)

Recommended Courses

- Programs that manipulate Programs
  - CIS 341: Compilers and interpreters
- Malware
  - CIS 331: Intro to Networks and Security
- Undecidability
  - CIS 262: Automata, Computability and Complexity



Recommended Reading


LOGICOMIX
CLICKCL
AN EPIC SEARCH
APOSTOLOS DOXIADIS, CHRISTOS
ALECOS PAPADATOS, AND AN
Secure Coding in C and C++
A CERT® BOOK
bert C. Seacord
word by Richard D. Pethia
CERT Director
Building Secure Software
How to Avoid Security Problems the Right Way
John Viega
Gary McGraw
Foreword by Bruce Schneier
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
I AM A STRANGE LOOP
LAS
DTER
WINNER OF THE PULITZER PRIZE
GÖDEL, ESCHER, BACH:
AN ETERNAL GOLDEN BRAID
DOUGLAS R. HOFSTADTER
A METAPHORICAL FUGUE ON MINDS AND MACHINES IN THE SPIRIT OF LEWIS CARROLL