Power of software

What’s going on

- User report grades posted
- 1.0 release due May 1, 11:55 PM
- Presentations April 30/May 1
  - managers will submit preference by Sunday
    - GeoMusic, Luper, and Pangea presented first last time
    - AutoAdvisor, Achieve!, and TeamAlphaStrikeForce’s second
- Final team assessment due May 1 (late OK)

Today’s plan

- Exam review
- Evaluations
- Power of computing
- Zero knowledge proofs

What’ll be on the exam?

- testing
- debugging
- working in groups
- reasoning about programs
  - [marker-red]

Testing

- Know about different kinds of tests
  - unit, integration, regression, etc.
- Know about different kinds of coverage
  - statement, path, etc.
- Know what’s hard about testing
  - GUI, usability, covering all behavior, etc.

Debugging

- Know four kinds of defense against bugs
  - make impossible
  - don’t introduce
  - make errors visible
  - last resort: debugging
- Rep invariants
- Assertions
Working in groups

• What’s hard?
  – corner cases
  – complete specification covers A LOT of behavior
  – unless a spec is concise, it’s hard to understand
  – precision is hard: language is ambiguous
  – communication is important

Reasoning about programs

• Ways to verify your code
  – testing, reasoning, proving
• Forward reasoning
• Backward reasoning
• Loop invariants
• Induction
• Practice some examples!

Loop example

Find the weakest precondition

```java
for (int x = 1; x <> y;) {
    if (y > x) {
        y = y / 2;
        x=2*x;
    }
}
// postcondition: x=8, y=8, and x and y are ints
```

you can also find the loop invariant and decrement function

What’s the test going to be like?

• 2 thinking questions
• the rest will be multiple choice
• will be designed to take less than an hour

On Tuesday, come with review questions and I will answer them.

Power of Computing

Can you write any program I describe to you?

Can you write:

A program HALTS? whose input is the body of a method, and that outputs 0 if the method enters an infinite loop, and 1 if it does not.
What’s HALTS?(method)?
method() {
    print “hello world”;
}

What’s HALTS?(method)?
method() {
    for (int x=0; x<5; x++)
        print “hello world”;
}

What’s HALTS?(method)?
method() {
    for (int x=0; x<-1; x++)
        print “hello world”;
}

What’s HALTS?(method)?
method() {
    while (true);
}

What’s HALTS?(method)?
method() {
    int x = 785th digit of π;
    if (x == 7)
        while(true);
}

What’s HALTS?(method)?
method() {
    int x = 785th digit of π;
    int y = x^x^x^x^x+1;
    int z = y10th digit of π;
    if (z == 0)
        while(true);
}
What’s HALTS?(method)?

```java
method() {
    int x = 785\textsuperscript{th} digit of \pi;
    int y = x\times x\times x\times x+1;
    int[] z[] = the \ y\textsuperscript{th} through (x+y)\textsuperscript{th} digits of \pi;
    if (z ever repeats in \pi again)
        while(true);
}
```

How about the general case?

• Let’s count programs. How many programs are there?
• And how many problems are there?
  – let’s limit ourselves to simple problems:
    * given a set of numbers, e.g., \{2, 4, 6\}
    * on input \(i\), return 1 if \(i\) is in the set, and 0 otherwise

First 64 programs

• How many of our problems can I solve with 64 programs?
  (a) 64
  (b) 32
  (c) 8
  (d) 6
  (e) 2

First 64 programs

• With 64 programs, how large can my sets get (if I am being compact)
  (a) 64
  (b) 32
  (c) 8
  (d) 6
  (e) 2
• Example: with 4 programs, I could cover:
  \{\}, \{1\}, \{2\}, \{1,2\}

Scalability Problem

• To cover subsets of a set of \(n\) numbers, I need \(2^n\) programs.
• But I only have as many programs are there are natural numbers.
• That’s exponentially smaller than the number of problems there are.

Can’t do it for all subsets!

Can HALTS? exist?

• Imagine that you wrote HALTS?
  • I will write a new program NALTS?:
    ```java
    NALTS?(Method p) {
        if (HALTS?(p)==0) return 1;
        else while (true);
    }
    ```
  • Key, run the program on (almost) itself

What is the value of NALTS?(NALTS?)
What is the value of NALTS? (NALTS?)

• Two cases:
  1. If NALTS?(NALTS?) goes into an infinite loop, then
     HALTS?(NALTS?)==1, which means that NALTS? terminates.
     So case 1 is impossible.
  2. If NALTS?(NALTS?) does not go into an infinite loop, then HALTS?(NALTS?)==0, which means that NALTS? does not terminate.
     So case 2 is impossible.

Conclusion

• The program HALTS cannot exist!
• Many programs cannot exist!
• Learn more in CS 601

Zero-Knowledge Proofs

How can I prove to you I know X without telling you anything about X?