Power of software

What’s going on
• User report are being graded
• 1.0 release due Wed April 29, 11:55 PM
• Presentations Mon April 27
• Final team assessment due May 1

Today’s plan
• Exam review
• Evaluations
• Power of computing

What’ll be on the exam?
• testing
• debugging
• working in groups
• reasoning about programs
• power of software (high-level questions only)

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
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

When and Where?

• Thursday May 7, 3:30 PM
• Hasbrouck Lab Add room 124
  C3 on http://www.umass.edu/visitorsctr/sites/default/files/maps/campus-map.pdf

Evaluations

• We’ll take 15 minutes to do evaluations
• They are anonymous and I don’t see them until (long) after the grades are posted
• I actually use them to improve my teaching
• UMass uses them to decide if I am a good teacher and whether to let me keep teaching
• UMass cares most about question 11, and also about questions 12 and 10

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)?

```java
method() {
    print “hello world”;
}
```

What’s HALTS?(method)?

```java
method() {
    for (int x=0; x<5; x++)
        print “hello world”;
}
```

What’s HALTS?(method)?

```java
method() {
    for (int x=0; x<-1; x++)
        print “hello world”;
}
```

What’s HALTS?(method)?

```java
method() {
    while (true);
}
```

What’s HALTS?(method)?

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

```java
method() {
    int x = 785\textsuperscript{th} digit of \pi;
    int y = x^{x^{x^{x^{x+1}}}};
    int z = y\textsuperscript{th} digit of \pi;
    if (z == 0)
        while(true);
}
```

What’s HALTS?(method)?

```java
method() {
    int x = 785\textsuperscript{th} digit of \pi;
    int y = x^{x^{x^{x^{x+1}}}};
    int[] z[] = \text{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

- 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\}

First 64 programs

- How many of our problems can I solve with 64 programs?
  (a) 64
  (b) 32
  (c) 8
  (d) 6
  (e) 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?:
  
  ```
  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 401 or CS 601

Zero-Knowledge Proofs

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