Debugging an Operation

Using a Failed Proof of Correctness as a Guide

Joan Krone – Denison University
Joseph Hollingsworth – Indiana University Southeast

What Went Wrong?

- Student implements an operation
- Student creates a testing harness
  a small client program
- Student runs the testing harness
- Receives unexpected output
- The operation doesn't seem to do anything!

Invert

```
Operation Invert (Q: P_Queue);
        ensures Q = Reverse(#Q);
```

The outgoing Q equals the reverse of the incoming Q

- #Q – incoming
- Q – outgoing

P_Queue is a preemptable queue

- Has one additional operation – Inject

String Theory

Used for mathematically modeling a Queue

- Γ - is the alphabet, e.g., $\Gamma = \mathbf{Z}$

String Theory

Used for mathematically modeling a Queue

- Γ - is the alphabet, e.g., $\Gamma = \mathbf{Z}$
- Example strings:
  - α = <4,22,3,17> *string of 4 integers*
  - β = < > *empty string*

String Theory

Used for mathematically modeling a Queue

- Γ - is the alphabet, e.g., $\Gamma = \mathbf{Z}$
- Example strings:
  - α = <4,22,3,17> *string of 4 integers*
  - β = < > *empty string*
- Concatenation, Length, Reverse
  - <-5> o α o <101> = <-5,4,22,3,17,101>
  - $|\alpha| = 4$
  - Reverse(α) = <17,3,22,4>
  - Reverse(α o <101>) = <101,17,3,22,4>

Testing Harness for Invert

```
Begin
      q: P_Queue;
      Enqueue(3, q);
      Enqueue(2, q);
      Enqueue(1, q);
      Writeln("q before Invert = ", q);
      Invert(q);
      Writeln("q after Invert = ", q);
end;
```

Output:

q before Invert = <3,2,1>

q after Invert = <3,2,1>

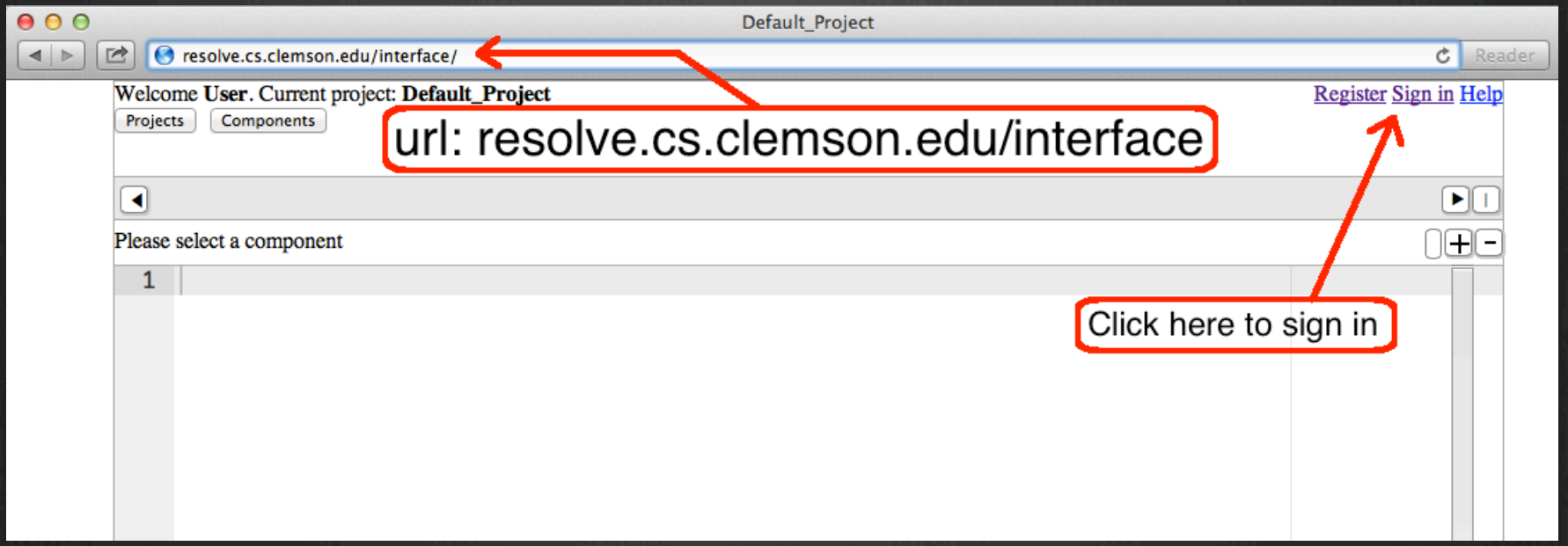
How to Debug?
I Could Try …

- Using one of those genetic algorithms to make random mutations to my code …

Or Maybe I Should …

- Do that proof of correctness thingy that my professor showed us.

Accessing the RESOLVE Web Interface



Accessing Components