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!
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

- $\Gamma$ - is the alphabet, e.g., $\Gamma = \mathbb{Z}$
String Theory

Used for mathematically modeling a Queue

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

Example strings:
- $\alpha = <4,22,3,17>$ string of 4 integers
- $\beta = <$ empty string
String Theory

Used for mathematically modeling a Queue

- $\Gamma$ - is the alphabet, e.g., $\Gamma = \mathbb{Z}$
- Example strings:
  - $\alpha = <4,22,3,17>$  string of 4 integers
  - $\beta = <>$  empty string
- Concatenation, Length, Reverse
  - $<-5> \circ \alpha \circ <101> = <-5,4,22,3,17,101>$
  - $|\alpha| = 4$
  - $\text{Reverse}(\alpha) = <17,3,22,4>$
  - $\text{Reverse}(\alpha \circ <101>) = <101,17,3,22,4>$
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

url: resolve.cs.clemson.edu/interface

Click here to sign in
Accessing Components

Click "Components" button to access component library and to add your own.

Components open up in their own tab.

```plaintext
1 Realization Defective_Recursive_Inverting_Realiz
2 for Inverting_Capability of Preemptable_Queue_Template;
3 Recursive Procedure Invert(updates Q: P_Queue);
4    decreasing |Q|;
5 Var E: Entry;
6 If (Length(Q) /= 0) then
7    Dequeue(E, Q);
8    Invert(Q);
9    Inject(E, Q);
10 end;
11 end Invert;
12 end Defective_Recursive_Inverting_Realiz;
```