Debugging

Debugging

- "**Debugging** is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system."
(https://en.wikipedia.org/wiki/Debugging, accessed January 25, 2019)

- **Reference material from Code Complete in Brightspace**

DALHOUSIE UNIVERSITY
Inspiring Minds

Debugging steps

- Stabilize the error
  - Ensure that it is reproducible
- Locate the source of the error
  - Gather the data that produces the defect
  - Analyze the data and form a hypothesis about the defect
  - Determine how to test the hypothesis
  - Prove or disprove the hypothesis
- Fix the defect
- Test the fix
- Look for similar errors

Finding Defects

- Use all data available to make your hypothesis
- Refine the test cases that produce the error
- Exercise the code in your unit test suite
- Use available tools
- Reproduce the error several different ways
- Generate more data to generate more hypotheses
- Use the results of negative tests
- Brainstorm for possible hypotheses
- Keep a list of things to try
- Narrow the suspicious region of the code
- Check common defects
- Take a break from the problem
- Set a maximum time for quick-and-dirty debugging
- Make a list of brute-force techniques and use them

Steps from "Code Complete" by Steve McConnell, 2004

Debugging Tactics

- Print statements
  - Not inherently bad, but too often used as the only tactic
- Bisecting code
- Tracing code
- Modifying the behavior of your program dynamically
  - Change variables
  - Run extra code to effect a change and see the effect
  - Jumping directly to another spot in the code
  - Re-running some code that just finished (and failed)
- Reviewing how you reached one point

DALHOUSIE UNIVERSITY
Inspiring Minds

Debuggers

- Environments in which to execute your program that also allows you to inspect, interact, and change the execution of the program dynamically

Navigating

- Breakpoints
  - Conditional breakpoints
  - Enable and disable
- Next / Step over
- Step / Step into
- Finish / Step return
- Continue
- Run to

Variables

- View variable values
  - Expand data structures
- Change values
- Automatically monitor for changes

Call stack

- Move up and down the call stack to
  - See what variables came in or out
  - See where you're coming from
- Restart the current execution at an earlier stack

Run new code

- Call methods to effect changes
- Write Java snippets to test conditions, help view values, or change values