CSci 3081W

Testing

Test and Debug *Effectively*

Questions

What do we mean by testing?

What are some effective testing practices?

Why is testing both overrated and underrated?

How are testing and debugging related?

Exercise, Part 1

Suppose you are working at a software development company, and your boss gives you some code to test (that you have not written), and asks you for a “test plan.”

Q1) What do you think a test plan is?

Q2) What would your test plan consist of?

See next slide for a more specific problem statement.

Exercise, Part 1

Suppose your boss asks you to create a test plan (for you and two co-workers) for the `Polyline` class described on the exercise description sheet: is the class error-free? Does it correctly implement the requirements and design? Etc. If you had to write a half-page email to your boss describing the test plan, what would you include? For example, what tasks would you include? And how would you distribute the work among your team?

Don't actually write the email, but outline it, or in some other way summarize what you would include in it.

Write your outline or summary on the exercise hand-in sheet.

Test Plan (Partial) Checklist

- Did your test plan include checking the code against the requirements?
- Did your test plan include checking the code against the design?
- Did you include a variety of types of testing?
- Did you identify any parts of the code that might need more than the usual amount of scrutiny?
- Did you assign specific tasks to specific team members?
- Did you assign some key activities as collaborative tasks rather than partition all the testing tasks to individuals?

McConnell Checklist (p. 532)

- Does each requirement that applies to the class or routine have its own test case?
- Does each element from the design that applies to the class or routine have its own test case?
- Has each line of code been tested with at least one test case? Has this been verified by computing the minimum number of tests necessary to exercise each line of code?
- Have all defined-used data-flow paths been tested with at least one test case?
- … (many more — see the textbook)

Iteration 2 (Quick) Self-Reflection

- Suppose as part of Iteration 2 you had to submit a report on your group's testing.
- What would you include in such as report?
- What types of testing did you and your group do?
- What types of testing was most effective?
- What testing would you like to improve?

What Is Testing?
(the "big picture")

Key Points

- Testing is a key step in software development.
- There are many, many, different types of testing.
- There are many different ways to categorize types of testing or testing activities.
- In a large software development organization, testing is done not only by the code writers, but also by others (some large companies have an entire division dedicated to testing).

(Some) Types of Testing

- System Validation: Did we write the right software?
- System testing
- Integration testing
- Unit testing
- Regression testing
- ..

(Some More) Types of Testing

- Black box
- White box
- Structural
- Perturbation
- User
- ... (some programs have an entire class just on testing)

Types of Testing

- See McConnell Ch. 22 for a discussion of these and other types of testing.
- We don't have time in this class for a comprehensive discussion of all these types of testing, but we'll focus on one type of structural testing: coverage testing, especially the structured basis testing in McConnell 22.3.

Coverage Testing

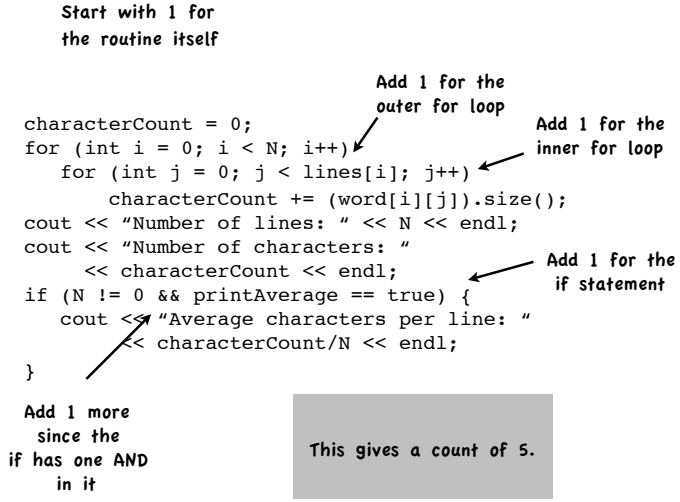
Suppose you have a set of test cases. Some questions you can ask:

1. Is every statement containing a *condition* executed by at least one test case?
2. Is every *statement* in the code executed by at least one test case?
3. Is every *branch* in the code covered in the following sense: is every condition evaluated to true by at least one test case, and to false by at least one test case?
4. Is each part of a *compound* condition evaluated to true by at least one test case, and to false by at least one test case?

Structured Basis Testing

Mentioned in McConnell, Ch. 22.3 (pp. 505 - 509). To find a minimal number of test cases to cover all possible values of each part of each condition in a of a piece of code:

1. Count 1 for the code itself.
2. Count 1 for every loop.
3. Count 1 for every if, else, etc.
4. If any condition (including loop termination) is compound, add one for each additional part of the condition.



Exercise Part 2

For each of the following three test sets, state whether the set covers each statement, condition, condition branch, or structured basis testing for the `Polyline::cornerCut()` code. Write your answers on the exercise hand-in sheet.

1. A single polyline with points (0, 0), (1, 0), (2, 0) and a minimum distance of .1
2. A single polyline with points (0, 0), (.9, 4), (1.0, 4), (1.1, 4), (2, 2), (2, 0) and a minimum distance of .2
3. Two polylines. The first with points (0, 0), (1, 0), (1.1, 1) and minimum distance of .2; and the second with points (0, 0), (0.1, 1), (2, 0), (3, 1) and a minimum distance of .2.

Exercise Part 3

What is the number of test cases that structured basis testing would need for the `Polyline::cornerCut()` code?

If none of the tests in Part 2 provided structured basis testing, then come up with a set of test cases that does.

Write your answers to the above on the exercise hand-in sheet.

Challenge: What constitutes a "test case" is not obvious here. Is the number of test cases in a test set here the number of polylines tested? The total number of points in the polylines? Neither?

One More Question

- Today's class has focused primarily on testing.
- What about debugging?

Testing and Debugging

"Debugging is the process of identifying the root cause of an error and correcting it. It contrasts with testing, which is the process of detecting the error initially. On some projects, debugging occupies as much as 50 percent of the total development time. For many programmers, debugging is the hardest part of programming." (McConnell, p. 535)

Introductory Exercise (Quick Reflection)

- Recall the Polyline class.
- Polyline::rotate() does not work correctly: applying this function seems to rotate the polyline the wrong amount. How would you go about finding the error causing this incorrect behavior? (Assume you have access to the relevant requirement, design, source code, and executable code.)

Table From McConnell (p. 537, from Gould 1975 reference)

	Fastest Three Programmers	Slowest Three Programmers
Average debug time (minutes)	5.0	14.1
Average number of defects not found	0.7	1.7
Average number of defects made correcting defects	3.0	7.7

Two Points

- Significant difference between most-skilled and least-skilled debuggers.
- Debugging is not a perfect process: even the most skilled debuggers missed some defects and introduced other defects when debugging the code.

Debugging Discussion Questions

Discuss the following questions with the person or persons next to you:

1. Why can debugging be difficult?
2. How do you normally debug code? For example, suppose you get a number of compiler errors. What is your usual way of finding and fixing those errors?
3. Suppose you are running a program and the output is obviously incorrect. What is your usual way of finding and fixing the underlying code errors?
4. Suppose your usual approach does not work. Then what do you try?

Debugging Reading

- McConnell Chapter 23 is about debugging.
- He contrasts sloppy debugging practices with more methodical, intentional practices.
- Some of what he says is common sense, but even these are good reminders.

A Few Tips From Ch. 23

- Be suspicious of classes and routines that have had defects before.
- Check for common defects.
- Fix the problem, not the symptom (this requires that you understand the root cause of the problem).
- Make one change at a time.
- Check your fix.

Some Summary Debugging Points

- Be methodical and careful. Don't use guessing or random changes as your go-to debugging strategy.
- Learn what debugging practices the best software developers do.
- Use a variety of techniques and tools.
- Friday's lab will be on using a debugger. If you have never used one before, then this is a chance to do so. If you have, this is a chance for further debugger practice.

End-of-Class Writing

Please write answers to the following two questions on the exercise hand-in sheet, and hand in that sheet at the end of class today.

1. What is one important thing you learned about testing and/or debugging today?
2. What is one thing you learned today from the discussions with others in the class? List one thing you found particularly interesting, important, and/or surprising.