

**SE 2AA4, CS 2ME3 (Introduction to Software  
Development)**  
**Winter 2017**

## **29 Introduction to Verification**

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 16, 2017



# Introduction to Verification

- Today's slide are partially based on slides by Dr. Wassyng, Ghezzi et al
- Administrative details
- `pointInRegion(p)`

# Administrative Details

- Investigating 9 academic integrity cases for A2
- A3 deadlines
  - ▶ Part 2 - Code: due 11:59 pm Mar 20
  - ▶ Part 1 spec available in repo
  - ▶ Change of  $<$  to  $\leq$  in natural language and spec
- A4
  - ▶ Your own design and specification
  - ▶ Due April 3 at 11:59 pm

# A Table for pointInRegion(p)

- Consider all of the cases
- Draw a picture
- Short form notation
  - ▶  $px = p.xcoord()$
  - ▶  $py = p.ycoord()$
  - ▶  $llx = lower\_left.xcoord()$
  - ▶  $lly = lower\_left.ycoord()$
  - ▶  $llxw = lower\_left.xcoord() + width$
  - ▶  $llyh = lower\_left.ycoord() + height$
  - ▶  $T = Constants.TOLERANCE$

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq T$
	$lly \leq py \leq llyh$	$(llx - px) \leq T$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq T$
$llx \leq px \leq llxw$	$py < lly$	$(lly - py) \leq T$
	$lly \leq py \leq llyh$	True
	$py > llyh$	$(py - llyh) \leq T$
$px > llxw$	$py < lly$	$p.\text{dist}(\text{PointT}(llxw, lly)) \leq T$
	$lly \leq py \leq llyh$	$(px - llxw) \leq T$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llxw, llyh)) \leq T$

# Seven Cases

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq T$
	$lly \leq py \leq llyh$	$(llx - px) \leq T$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq T$
$llx \leq px \leq llxw$		$(lly - T) \leq py \leq (llyh + T)$
$px > llxw$	$py < lly$	$p.\text{dist}(\text{PointT}(llxw, lly)) \leq T$
	$lly \leq py \leq llyh$	$(px - llxw) \leq T$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llxw, llyh)) \leq T$

# Six Cases

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq T$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq T$
$llx \leq px \leq llxw$		$(lly - T) \leq py \leq (llyh + T)$
$px > llxw$	$py < lly$	$p.\text{dist}(\text{PointT}(llxw, lly)) \leq T$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llxw, llyh)) \leq T$
$lly \leq py \leq llyh$		$(llx - T) \leq px \leq (llxw + T)$

# Three Cases

	out
$llx \leq px \leq llxw$	$(lly - T) \leq py \leq (llyh + T)$
$lly \leq py \leq llyh$	$(llx - T) \leq px \leq (llxw + T)$
$\neg(llx \leq px \leq llxw) \wedge \neg(lly \leq py \leq llyh)$	$\min[p.\text{dist}(\text{PointT}(llx, lly)),$ $p.\text{dist}(\text{PointT}(llxw, lly)),$ $p.\text{dist}(\text{PointT}(llx, llyh)),$ $p.\text{dist}(\text{PointT}(llxw, llyh))] \leq$ $T$



# Nine Cases, but 2D

- How would you write all 9 cases, but with a tabular form that closely matches the original 2D problem description?

# Outline of Verification Topics

- What are the goals of verification?
- What are the main approaches to verification?
  - ▶ What kind of assurance do we get through testing?
  - ▶ How can testing be done systematically?
  - ▶ How can we remove defects (debugging)?
- What are the main approaches to software analysis?
- Informal versus formal analysis

# Testing on Assignment 1 to 4

- Limited guidance on test case selection
- Improved test cases would have improved the results
- Consider the method for deleting from a sequence of T (next slide)
- We are moving toward automated testing
- We have seen the advantages of regression testing
- Some have adopted the excellent strategy of test as you develop
  - ▶ Helps isolate errors
  - ▶ Does not leave testing to the end when there is no time to do it properly
  - ▶ Helps improve the understanding of the problem and the program
- Hopefully the experience on the assignments has motivated you to think more about testing

## Incorrect Version of Delete

Using `s = new T[MAX_SIZE]`, for some type `T`

```
public static void del(int i)
{
    int j;

    for (j = i; j <= (length - 1); j++)
    {
        s[j] = s[j+1];
    }

    length = length - 1;
}
```

What test cases will highlight the error?

## Correct Version of Delete

```
public static void del(int i)
{
    int j;

    for (j = i; j < (length - 1); j++)
    {
        s[j] = s[j+1];
    }

    length = length - 1;
}
```

Avoids potential `ArrayIndexOutOfBoundsException` Exception

# Need for Verification

- Designers are fallible even if they are skilled and follow sound principles
- We need to build confidence in the software
- Everything must be verified, every required functionality, every required quality, every process, every product, every document
- Even verification itself must be verified

# Properties of Verification

- May not be binary (OK, not OK)
  - ▶ Severity of defect is important
  - ▶ Some defects may be tolerated
  - ▶ Our goal is typically acceptable reliability, not correctness
- May be subjective or objective - for instance, usability, generic level of maintainability or portability
- Even implicit qualities should be verified
  - ▶ Because requirements are often incomplete
  - ▶ For instance robustness, maintainability