



UMassAmherst

Manning College of Information  
& Computer Sciences

Programming Methodology

# Lab 6: Property-Based Testing and Object Oriented Programming

Wednesday October 8, 2025

# Weekly Lab Agenda

- Go over reminders/goals
- Review past material
- Work in groups of 2-3 to solve a few exercises
  - Please sit with your group from last week.
- Discussion leaders will walk around and answer questions
- Solutions to exercises will be reviewed as a class
- Attendance taken at the end

# Reminders

- Homework 4a is due Sunday (10/12) at 11:59pm
  - Come to office hours for help!
- Homework 4b is due the following Sunday (10/19)
- Midterm 1 is being graded. Aiming to finish grading by Friday.
  - Exam solutions will also be posted later this week.
- Submit what you have at the end of lab to Gradescope. If you miss many submissions to Gradescope, we may penalize your lab grade.

# Today's Goals

- Property-Based Testing
- Object Oriented Programming

# Property-Based Testing

Used when a problem has **more than one** right answer.

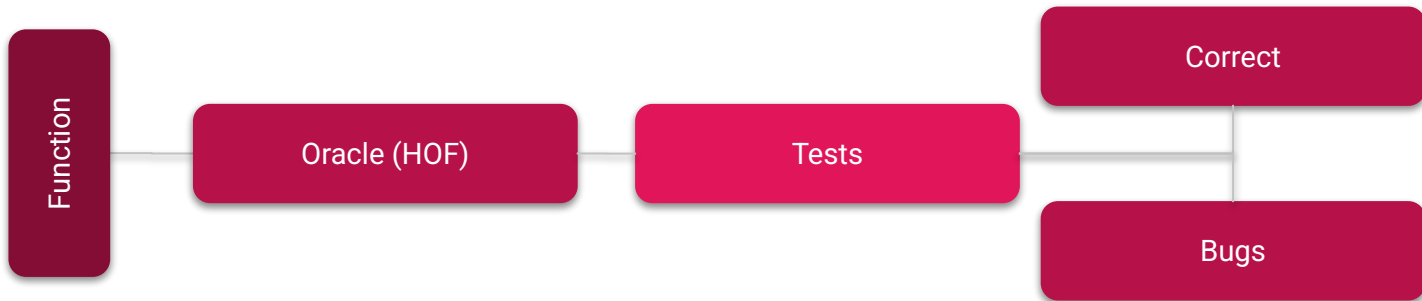
Steps to Property-Based Testing:

1. Start with a valid input to your problem
2. Run the algorithm on that input
3. Check that the result has all necessary characteristics

# Oracle Functions

In this class, **oracles** are functions that determine whether or not a function correctly solves a given problem. We can determine this by performing property based testing on the outputs of the given function for various inputs.

For example, in homework 4a, you write an oracle for the Stable Matching Problem.



# Exercise 1: Permutations

A function **genArray**( $n$ : number): number[][] is supposed to generate an  $n \times n$  array of numbers such that each row and column is a permutation of the numbers from 0 to  $n-1$ . Assume  $n$  is nonnegative.

**Write an oracle that accepts the function genArray as input.**

Use higher-order functions when appropriate. Try to write your implementation in  $O(n^2)$ .

# Review: Rectangle Class

In lecture, we explored drawing on a canvas as an example of classes and objects. For example, we looked at the Rect class, which represented a rectangle on a canvas.

Note that Rect was written in JavaScript, not TypeScript.

**What changes do we need to make for TypeScript?**

```
class Rect {  
  constructor(x, y, w, h) {  
    this.loc = { x: x, y: y }; // top left corner  
    this.width = w;  
    this.height = h;  
  }  
  draw(ctx, color) {  
    const x = this.loc.x;  
    const y = this.loc.y;  
    const w = this.width;  
    const h = this.height;  
    drawLine(ctx, x, y, x + w, y, color);  
    drawLine(ctx, x + w, y, x + w, y + h, color);  
    drawLine(ctx, x + w, y + h, x, y + h, color);  
    drawLine(ctx, x, y + h, x, y, color);  
  }  
}
```



# Review: Rectangle Class

In TypeScript, we must first declare class properties! We also need to add type annotations, as you are familiar with.

```
class Rect {  
    loc: { x: number; y: number };  
    width: number;  
    height: number;  
  
    constructor(x: number, y: number, w: number, h: number) {  
        this.loc = { x: x, y: y };  
        this.width = w;  
        this.height = h;  
    }  
  
    // drawLine does not change  
}
```

## Exercise 2: OOP

Implement the **Circle** class. Similar to the Rect class, the Circle class should have a property **loc** representing the x, y position of the circle. It should also have a **constructor** and a **draw** method.

Hint: you may want to use the drawCircle function, which you can find in the lab.ts file.

```
class Circle {  
  constructor(x: number, y: number, r: number) {  
    // TODO  
  }  
  draw(ctx: SKRSContext2D, color: string) {  
    // TODO  
  }  
}
```

Note: there is no autograder for this exercise.

Try testing your class by creating a Circle at x=80, y=70, and with radius 50. Then draw it using drawShape!