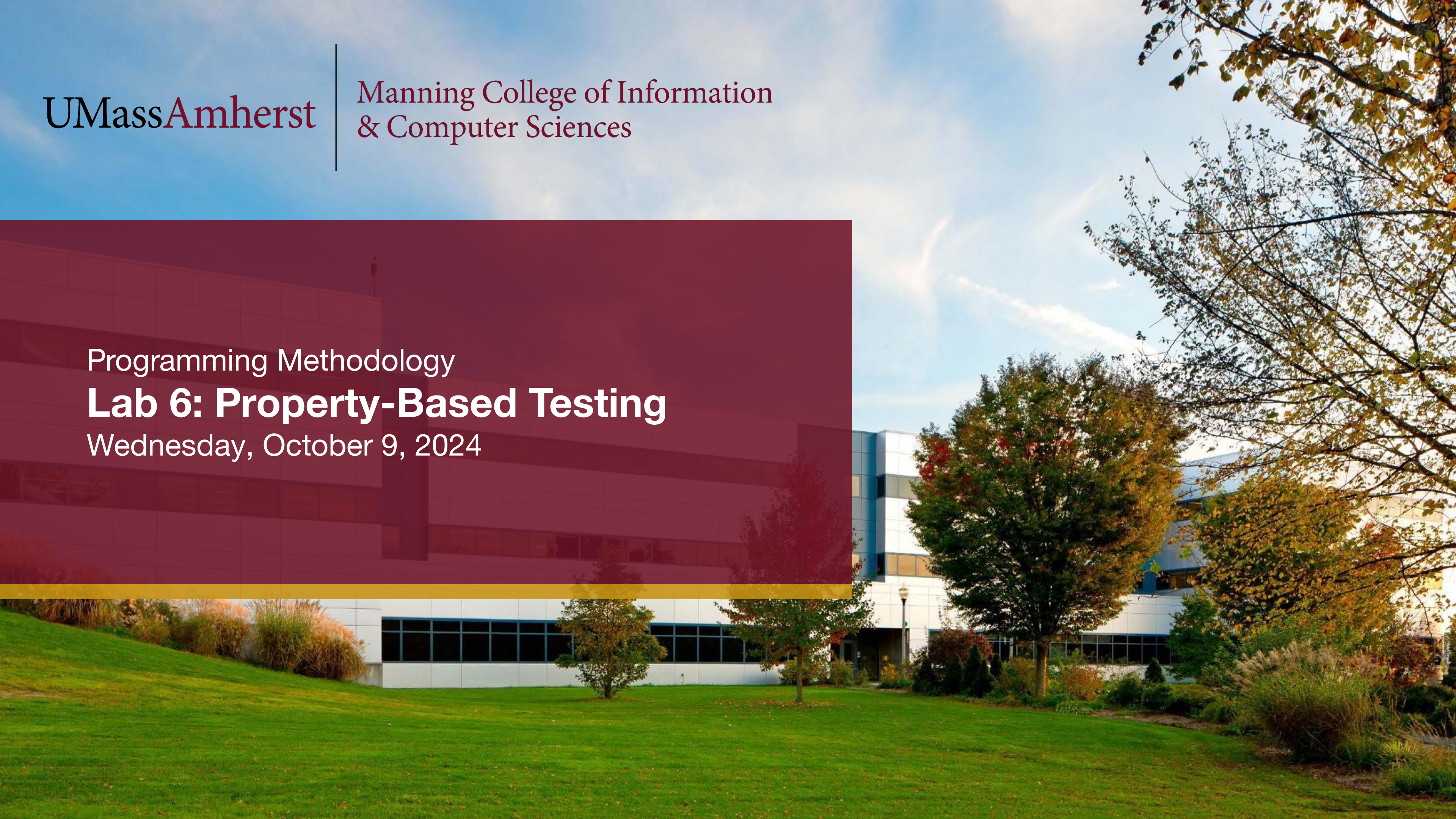


UMassAmherst | Manning College of Information & Computer Sciences

Programming Methodology

Lab 6: Property-Based Testing

Wednesday, October 9, 2024

Property-Based Testing

UMassAmherst | Manning College of Information & Computer Sciences

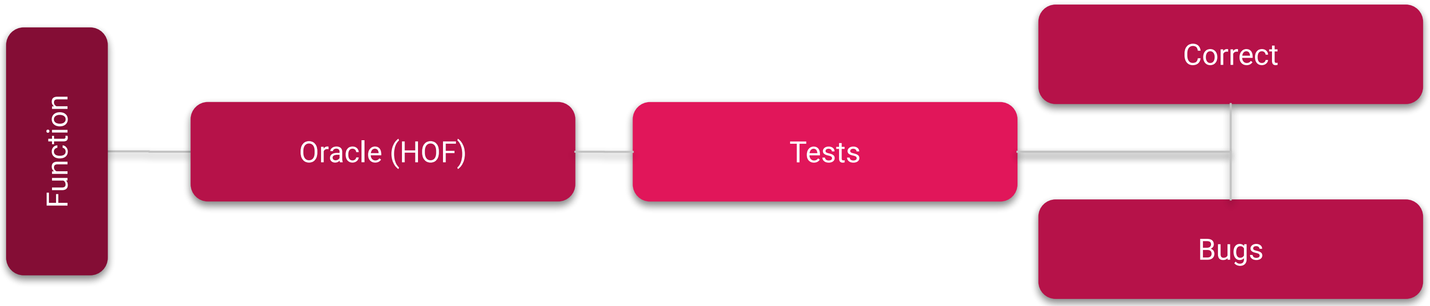
Used when a problem has **more than one** right answer

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

UMassAmherst | Manning College of Information & Computer Sciences



Exercise: Permutations

A function **genArray**(n: number): number[][] is supposed to generate an n × 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 $O(n^2)$.

Exercise: OOP

UMassAmherst | Manning College of Information & Computer Sciences

Think back to lecture where you discussed the shapes classes.
Before we start with this exercise, please familiarize yourself yourself for a few minutes with the code in the starter code. It should seem familiar to you.

Uncomment and run the three examples and make sure you understand!

- Your TA will demonstrate this.

Implement a **class** Translate whose constructor takes a shape and a change in x and change in y value. When the draw method is called with a CanvasRenderingContext2D (ctx) and a color, shift the canvas by dx and dy using ctx.translate, draw the shape on the moved canvas, and move the canvas back to the starting position.

Now run `npm run start`. You'll know your code is correct by the image.

Review Exam Questions

With the left over time we'll go over exam questions that you all found difficult.

Suggestion: Q4 or Q6, but we can go over any you want.

Midterm 1 - Q4

UMassAmherst | Manning College of Information & Computer Sciences

Write a function Q4 trackTemp(val: number): (diff: number) => [temp: number, days: number] that takes as argument the noon temperature on day 0 and returns a closure that can be called repeatedly. A call represents a new day, with a change in the noon temperature by an amount diff (positive, negative or zero) from the day before. The closure should return a pair of numbers: the noon temperature on the current day, and the number of consecutive days (≥ 1) that have passed without the temperature change diff having switched sign (zero is not a sign switch).

```
function trackTemp(val: number): (diff: number) => [temp: number, days: number] {
  let days = 0, lastS = 0;
  return diff => {
    if (lastS * diff < 0) days = 0;      // sign change
    if (diff !== 0) lastS = diff;        // try to make lastS <> 0, or keep it
    return [val+=diff, ++days];
  }
}
```

What are some relevant test scenarios?
Explain input, output, their purpose and what behavior they are testing.

Midterm 1 - Q6

UMassAmherst | Manning College of Information & Computer Sciences

An array represents employee wage increases in the years since their union was founded. There is one entry per year, representing the increase from the previous year as a ratio (assumed > 1). The first array entry is the increase in the year following the founding year. Write a function calcIncrease(byYear: number[]): ((wage: number) => number)[] that takes such an array and returns an array of closures, with the same length. When called with a wage amount for the founding year, the closure for each year will return the adjusted wage in that year, after all increases. Use reduce, no loops. Avoid needless repeated computations.

```
type numFun = (x: number) => number

function calcIncrease(byYear: number[]): numFun[] {
  const res: numFun[] = [];
  byYear.reduce((acc, rate) => {
    acc *= rate;
    res.push(x => x * acc);
    return acc;
  }, 1);
  return res;
}
```