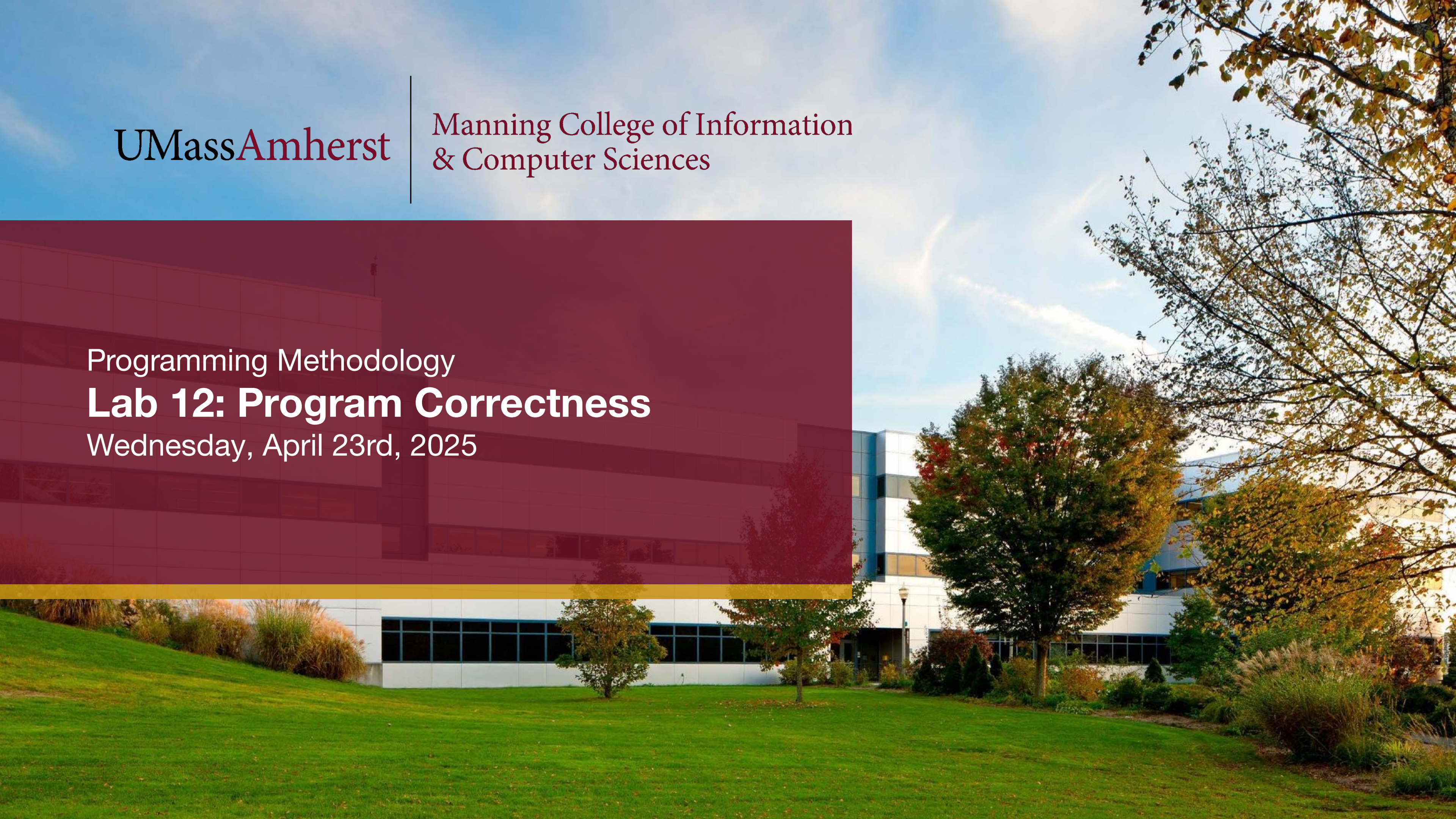

UMassAmherst | Manning College of Information & Computer Sciences

Programming Methodology

Lab 12: Program Correctness

Wednesday, April 23rd, 2025

Weekly Lab Agenda

- Go over reminders/goals
- Review past material
- Work in groups of 2-3 to solve a few exercises
    - Lab leaders will assign new groups this week
- Discussion leaders will walk around and answer questions
- Solutions to exercises will be reviewed as a class
- Attendance taken at the end

Reminders

- You should be working on your group project.

- Have a planned timeline to avoid last minute rush!

- Reach out to us if there is any issues in your team.
  - Private post on campuswire or
  - Email to mkuechen@umass.edu

UMassAmherst | Manning College of Information & Computer Sciences

Today’s Goals

- Practice working on program correctness.

UMassAmherst | Manning College of Information & Computer Sciences

What’s an Invariant?

- As its name says, an invariant is something that *does not change*
    - Here: a *specific type of assertion* (not all assertions are invariants)
- We work with **loop invariants**: they hold **before** each loop iteration (before checking the condition)
    - A useful invariant relates variables that *change* within the loop
    - The invariant will not hold at *every* point in the loop
      but will be *restored* (holds again) at the end

- In OOP, we also work with *method invariants* and *class invariants*

Binary Search Invariant

- Recall the case study on binary search we saw in class.
- What will be the invariants in this case:
  - a[lo-1] < x && x < a[hi+1] // x is not outside searched interval (we don't miss it)
  - (lo === 0 || a[lo-1] < x) && (hi + 1 === a.length || x < a[hi+1]) // ensure indices are valid

$$\frac{\{\, I \wedge C \,\}\ S\ \{\, I \,\}}{\{\, I \,\}\ \text{while}\ (C\,)\ S;\ \{\, I \wedge \neg C \,\}}$$

we only execute S (loop body) if C is true

at the end, the loop condition is false

$I \wedge \neg C$ holds (should imply what we want/need)

Question 1

The code is for binary search checking middle first.

This code has a bug, which can be identified by observing the invariant.

Use the rules to check if the invariant is maintained, or if there is some path through the code that fails to re-establish it.

```
function bsearchmid (a, target) {
    let lo = 0; let hi = a.length - 1;
    /* (lo === 0 || a[lo-1] < target)
           && (hi + 1 === a.length || target < a[hi+1]) */
    while (lo <= hi) {
     let mid = lo + Math.floor((hi-lo)/2);
         if (a[mid] == target) {
          return mid;
         } else if (a[mid] < target) {
          lo = mid;
         } else {
          hi = mid;
         }
    }
    return -1
}
```

Question 2

Please write your solution to question 2 on paper. You will submit your work to gradescope before we review the solution, and we'll grade your work to give you feedback.

Question 2

UMassAmherst | Manning College of Information & Computer Sciences

What is the largest value A for which the loop below terminates with n=20?

Use an invariant to justify your answer.

#Fall 2022 Final Exam

```
let n: number = 0;

let s: number = A;

while (s <= 100) {

     if (n % 2 > 0) {

        s += n; }

    n = n + 1;

}
```