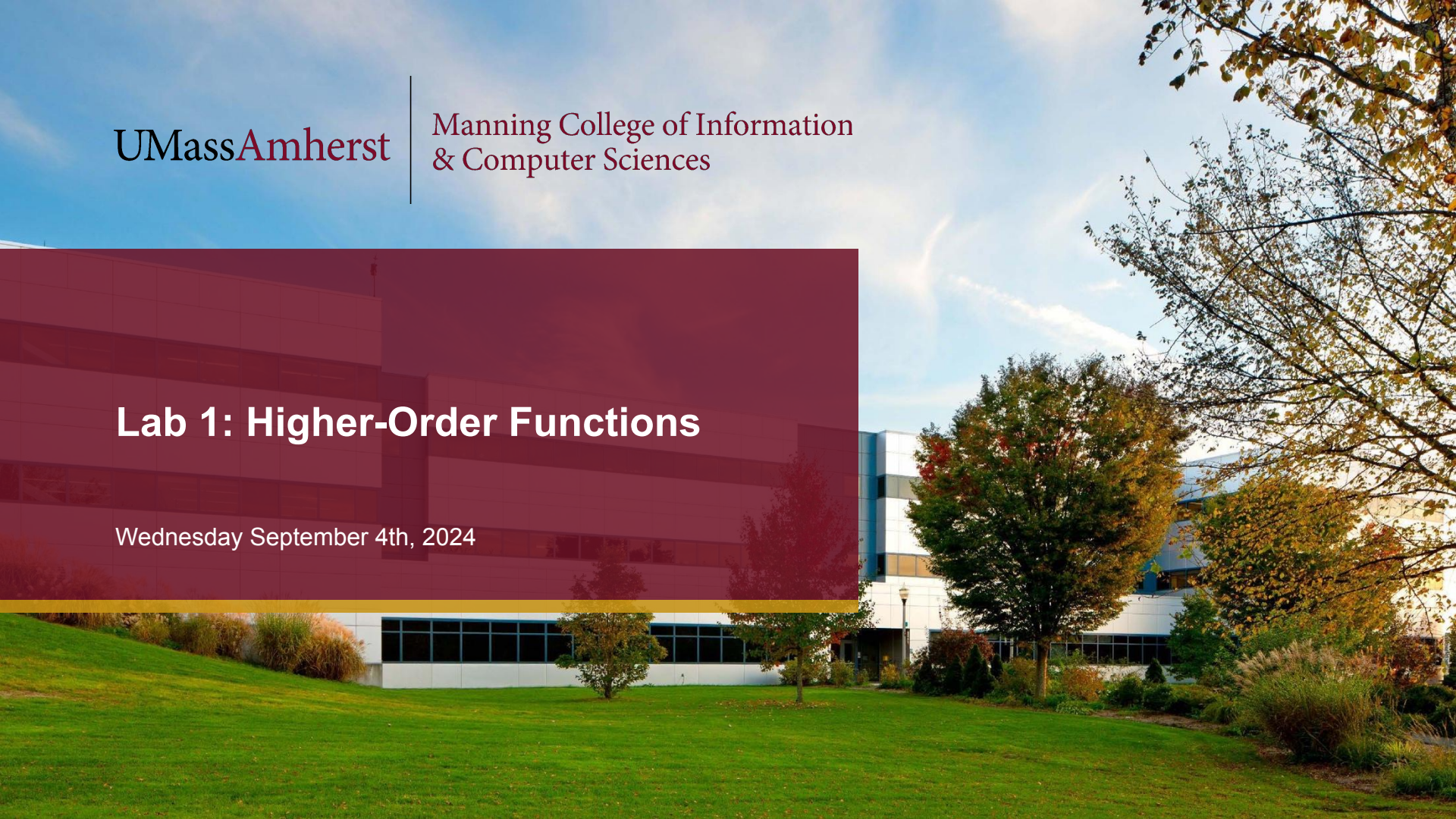


Lab 1: Higher-Order Functions

Wednesday September 4th, 2024

Weekly Lab Agenda

- Go over reminders/goals
- Review past material
- Work in groups of 2-3 to solve a few exercises
    - Discussion leaders will assign groups today
    - Groups will be remade every third lab
- Discussion leaders will walk around and answer questions
- Solutions to exercises will be reviewed as a class
- Attendance taken at the end

Reminders

- Please set up your development environment as soon as possible
  - “Setting up Your Development Environment” (Link on Canvas)
- Homework 1 has been released on Canvas
- If you need to miss lab and have a valid reason according to the syllabus (medical, other personal) please fill out the questionnaire on Canvas before the start time of your lab.
    - Waking up late, bus was late are NOT valid reasons to miss lab.
- Submit what you have at the end of lab to Gradescope. If you miss many submissions to Gradescope, we may penalize your lab grade.

Lab Groups

- You will now be assigned into your lab groups
- Please sit with your group each week

- Take the next 5 minutes to talk to each other
- Introduce yourselves!
  - Name, pronouns, major
  - Favorite household appliance
  - What was one fun thing you did over break?

Today's Goals

- Set up coding environment
- Practice both higher-order functions and some TypeScript
- Walk out with some working code

Writing and Running TypeScript

- Download the starter code from GitHub (linked on Canvas).
- Unzip the folder and open it in VSCode.
- Run *npm install* in the same directory as the package.json folder.
- When you are ready to submit, run *npm run build:submission*
- Upload the resulting zip file to the corresponding assignment on gradescope.

- Your lab leaders will walk you through this process for the first lab!

Review of map

```
// Sample Implementation
// `.map` is a method on Arrays
function map<T, U>(
  a: T[],
  f: (x: T) => U
): U[] {
  const result: U[] = [];
  for (let i = 0; i < a.length; ++i) {
     result.push(f(a[i]));
  }
  return result;
}
```

```
function double(x: number): number {
     return 2 * x;
}

const array = [1,2,3,4,5];
const newArray = array.map(double);
```

What is `newArray` ?

Reason about the code before typing and running it.

Review of filter

```
// Sample Implementation
// `.filter` is a method on Arrays
function filter<T>(
  a: T[],
  f: (x: T) => boolean
): T[] {
  const result: T[] = [];
  for (let i = 0; i < a.length; ++i) {
     const x = a[i];
     if (f(x)) {
       result.push(x);
     }
  }
  return result;
}
```

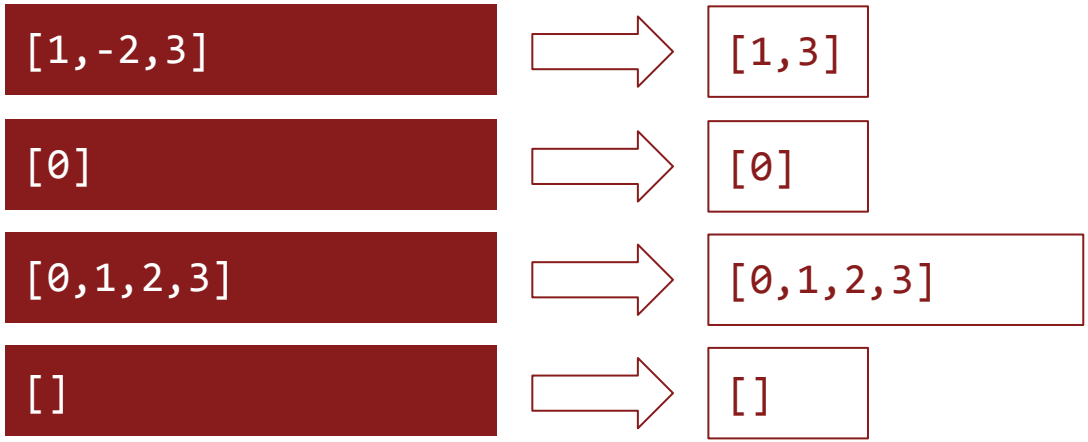
```
function isEven(x: number): boolean {
     return x % 2 === 0;
}
const array = [1,2,3,4,5];
const newArray = array.filter(isEven);
```

What is `newArray` ?

Reason about the code before typing and running it.

Programming Exercise 1

Without using any loops or recursion, write a function that takes in an array, and returns those elements of the array that are not negative.



Programming Exercise 1

Without using any loops or recursion, write a function that takes in an array, and returns those elements of the array that are not negative.

Solution 1 - As a separate named function:

```
const arr = [1, -2, 3]

function isNotNegative(x: number): boolean {
    return x >= 0;
}

console.log(
     arr.filter(isNotNegative)
);
```

Solution 2 - With arrow notation:

```
console.log(
     arr.filter((x: number) => x >= 0)
);
```

No types needed, compiler understands that x has to be a number. Could be simplified to:

```
console.log(
     arr.filter(x => x >= 0)
);
```

Programming Exercise 2

Using the code you wrote in exercise 1, write a function that takes an array of number arrays, and returns an array of number arrays where all negative values have been removed. Again, don't use any loops or recursion.

```
[[1,-2,3], [0], [0,1,2,3], []]
```

```
[[1,3],[0], [0,1,2,3], []]
```

A number array is typed as `number[]`, an array of those is `number[][]`

Programming Exercise 2

```
function keepNonNegativeValues(a: number[]): number[] {
  return a.filter(x => x >= 0);
}

const numberTable = [[1,-2,3], [0], [4,5,6], []];
console.log(
      numberTable.map(keepNonNegativeValues)
);
```

Important item here is to use the code developed in exercise 1. This is a good example of decomposing a problem into smaller problems, and of testing the smaller solutions before continuing.

Programming Exercise 3

Write a function that takes as input an array of number arrays, and returns an array of number arrays where any of the original arrays that had any negative values has been substituted with an empty array. Again, don't use any loops or recursion in our own code.

```
[[1,-2,3],[0],[4,5,6]]
```

```
[[],[0],[4,5,6]]
```

Programming Exercise 3

One possible approach: filter out the elements of the array that are negative. If the resulting array has the same number of elements as the original, put it in the result. Otherwise, put an empty array.

```
[[1,-2,3],[0],[4,5,6]]
```

```
[[],[0],[4,5,6]]
```

It's important to have this algorithm before starting to code.
Now we can use as much of the code we already wrote as possible.

Programming Exercise 3

```
const numberTableTwo = [[1,-2,3], [0], [4,5,6]];
console.log(
      numberTableTwo.map(keepNonNegativeRowOrEmpty)
);

function keepNonNegativeRowOrEmpty(A: number[]): number[] {
  return allNonNegative(A) ? A : [];
}

function allNonNegative(A: number[]): boolean {
  const filtered = keepNonNegativeValues(A); // Exercise 2
  return filtered.length === A.length; // If no values were dropped
}
```

Final Thoughts

- Think carefully about your approach before starting to code
- Start small and test often
- Try to make use of as much of your previous work as possible

- A lot of material all at once, come to office hours if you are confused
  - We would love to see you there! 😄
- It is okay if some of us take longer or don't finish before lab ends, keep working at it
  - Try not to get discouraged
- Starting the homework today is not required, but try to get into the habit of reading the instructions as soon as they are released