UMassAmherst | Manning College of Information & Computer Sciences

Programming Methodology

Lab 4

Wednesday, February 26, 2025

UMassAmherst | Manning College of Information & Computer Sciences

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 4 is due tonight at 11:59pm
    - Come to office hours for help!
- HW5 will not be released until after the exam.
- Midterm 1 is in one week on March 5th (Wednesday)
    - Start studying early. See past exams and solutions on Canvas.
-

UMassAmherst | Manning College of Information & Computer Sciences

Today’s Goals

- Iterators
- Mental models

UMassAmherst | Manning College of Information & Computer Sciences

Exercise 1: Iterators

- Write a function that takes two iterators over the same type and returns an iterator which will first exhaust iterator 1, then continue with iterator 2.

```
// concatenate two iterators
function concatIt<T>(it1: MyIterator<T>, it2: MyIterator<T>) {
   // your code here
}
```

Exercise 2

For each line of code: If a value is printed, state the value and describe how it was obtained, including any values used for the result. Otherwise, state what objects (including arrays, not closures or functions) are created (if any), what values are modified and which objects are no longer referenced (if any).

```javascript
const mkList = (init, f) => ({
  next: () => mkList(f(init), f),
  value: () => init
});
let cnt = 0;
const a = [mkList(0, x => x + 1), mkList(cnt, _ => ++cnt)];
const b = a.map(lst => lst.next().next());
console.log(b[1].value());
```

Exercise 3: More Iterators

- Write a function that prepends a value to an iterator. That is, return an iterator that will first produce the given value, then continue with the given iterator.

```
function prependIt<T>(v: T, it2: MyIterator<T>): MyIterator<T> {
     // TODO: Complete this function
}
```

Bonus Problem 4: Lists

Consider the following code fragment working with lists as defined in class.
How many list nodes (created with node()) are no longer accessible at the end of this code fragment?

```
let lst1 = ... // create a list with 2 elements
const concat =
  (l1, l2) => l1.isEmpty()? l2 : node(l1.head(), concat(l1.tail(), l2));
lst1 = concat(concat(lst1, lst1), lst1)
// end of the code fragment
```

Hints:
node constructor creates an object of of type List<T> and returns a reference to it.
Every call to the node constructor or to empty() creates a new object in memory.
At the end of the code, we have one variables in value map: lst1.
Objects that are not accessible through lst1 are no longer accessible.