CSE 331
Software Design & Implementation

Autumn 2023

Section 8 – Midterm Review

Administrivia

- Midterm
  - **Tomorrow at normal lecture times and locations**
  - Please arrive a couple minutes early so we can hand out exams and start right on time!
  - No notecards, all needed definitions will be included

- Midterm review session
  - **Tonight, 6-7pm**
  - **CSE1** (Allen), across breakout rooms
  - Announcement will be updated with details
  - Bring questions related to practice exams or general concepts

- HW 8
  - Released Monday, 11/20 (take a break this weekend ☺)
  - Due Wednesday 11/29, 11pm

Midterm review question topics

- **Reasoning about Recursion**
  - proving properties of recursive functions
- **Reasoning about Loops**
  - checking correctness of complex loops
- **Writing Methods**
  - correctly implementing methods of a class
- **Testing**
  - testing functions or methods using our heuristics

Definitions

The function at(n, S), returning the element at *index* n in the list S, is defined as follows:

$$\begin{aligned}
&\textbf{func}\ \mathsf{at}(n,\ \mathsf{nil}) &&:= \text{undefined} && \text{for any } n : \mathbb{N}\\
&\qquad \mathsf{at}(0,\ \mathsf{cons}(x, L)) &&:= x && \text{for any } x : \mathbb{Z} \text{ and } L : \text{List}\\
&\qquad \mathsf{at}(n+1,\ \mathsf{cons}(x, L)) &&:= \mathsf{at}(n, L) && \text{for any } n : \mathbb{N},\ x : \mathbb{Z} \text{ and } L : \text{List}
\end{aligned}$$

The function echo(S), returning a list with each element is repeated twice, is defined as:

$$\begin{aligned}
&\textbf{func}\ \mathsf{echo}(\mathsf{nil}) &&:= \mathsf{nil} &&\\
&\qquad \mathsf{echo}(\mathsf{cons}(x, L)) &&:= \mathsf{cons}(x, \mathsf{cons}(x, \mathsf{echo}(L))) && \text{for any } x : \mathbb{Z} \text{ and } L : \text{List}
\end{aligned}$$

The following two properties of the "at" function will be useful to us:

- **Fact A**: at$(2m, \text{echo}(S)) =$ at(m, S) for any $m : \mathbb{N}$ and S : List.
- **Fact B**: at$(2m+1, \text{echo}(S)) =$ at(m, S) for any $m : \mathbb{N}$ and S : List.

Familiar functions

The function len(L) returns the length of the list L:

$$\begin{aligned}
&\textbf{func } \mathsf{len}(\mathsf{nil}) && := 0 \\
&\qquad \mathsf{len}(\mathsf{cons}(x, L)) && := \mathsf{len}(L) + 1 && \text{for any } x : \mathbb{Z} \text{ and } L : \mathsf{List}
\end{aligned}$$

The function rev(L) returns a list containing the values of L in reverse order:

$$\begin{aligned}
&\textbf{func } \mathsf{rev}(\mathsf{nil}) && := \mathsf{nil} \\
&\qquad \mathsf{rev}(\mathsf{cons}(x, L)) && := \mathsf{concat}(\mathsf{rev}(L), \mathsf{cons}(x, \mathsf{nil})) && \text{for any } x : \mathbb{Z} \text{ and } L : \mathsf{List}
\end{aligned}$$

The function concat(S, R) returns a list containing the values of S followed by those of R:

$$\begin{aligned}
&\textbf{func } \mathsf{concat}(\mathsf{nil}, R) && := R && \text{for any } R : \mathsf{List} \\
&\qquad \mathsf{concat}(\mathsf{cons}(x, L), R) && := \mathsf{cons}(x, \mathsf{concat}(L, R)) && \text{for any } x : \mathbb{Z} \text{ and } L, R : \mathsf{List}
\end{aligned}$$

Problem 1

Fact A: $\mathsf{at}(2m, \mathsf{echo}(S)) = \mathsf{at}(m, S)$ for any $m : \mathbb{N}$ and S : List.

Prove **Fact A** by induction on **S**.

Hint: since the definition of “at” has a separate cases for $m = 0$ and $m = n + 1$, you may need to separate your induction step into those case as well.

ADT

```
/** A cursor is a list of integers. */
export interface IntCursor {
  /** @returns obj */
  values: () => List<number>;

  /** @returns at(n, obj) */
  valueAt: (n: number) => number | undefined;
}
```

ADT

```
class EchoCursor implements IntCursor {
  // AF: obj = echo(this.vals)
  readonly vals: List<number>;  // list before echo-ing

  constructor(vals: List<number>) {
    this.vals = vals;
  }

  // ... methods implemented later ...
}
```

```
/** @returns a cursor representing the list echo(vals) */
export const makeEchoCursor = (vals: List<number>): IntCursor => {
  return new EchoCursor(vals);
};
```

Problem 2a

```
values = (): List<number> => {
  let R: List<number> = this.vals;
  let S: List<number> = nil;
  {{ ______________________________________________ }}
  {{ Inv: echo(this.vals) = concat(rev(S), echo(R)) }}
  while (R !== nil) {
    {{ ______________________________________________ }}
    S = cons(R.hd, cons(R.hd, S));
    {{ ______________________________________________ }}
    R = R.tl;
  }
  {{ echo(this.vals) = rev(S) }}
  return rev(S);
};
```

Problem 2b

- Proving necessary implications where assertions meet by calculation
- Allowed to use "Because I said so" rule on **1 *calculation*** in part b, if you *don't* use it, we'll drop the lowest score

$$\begin{aligned}&\mathsf{echo}(\mathsf{this.vals})\\ &\quad = \mathsf{concat}(\mathsf{rev}(S), \mathsf{echo}(R)) \qquad \textbf{Because I said so}\end{aligned}$$

Problem 2b

I. Prove that the first assertion you filled in implies that **Inv** holds initially.

(Note: the def of concat is on the final page for reference.)

$$\{\{\ R = \text{this.vals and } S = \text{nil}\ \}\}$$
$$\{\{\ \textbf{Inv:}\ \text{echo}(\text{this.vals}) = \text{concat}(\text{rev}(S), \text{echo}(R))\ \}\}$$

Problem 2b

II. Prove that **Inv** and the fact that R = nil imply that the postcondition holds. You Lemma 2 in your calculation:
   - Lemma 2: concat(L, nil) = L for any list L

$$\{\{\ \mathsf{echo}(\mathsf{this.vals}) = \mathsf{concat}(\mathsf{rev}(S), \mathsf{echo}(R))\ \textbf{(Inv)}\ \text{and}\ R = \mathsf{nil}\ \}\}$$
$$\{\{\ \mathsf{echo}(\mathsf{this.vals}) = \mathsf{rev}(S)\ \}\}$$

Problem 2b

III. Prove that **Inv** and the fact that R ≠ nil imply that the assertion you filled in at the top of the loop body holds.

Use these Lemmas in your calculation:

- Lemma 3: rev(rev(L)) = L for any list L
- Lemma 4: concat(concat(L, R), S) = concat(L, concat(R, S)) for any lists L, R, S.

$$\{\{\ \mathsf{echo}(\mathsf{this.vals}) = \mathsf{concat}(\mathsf{rev}(S), \mathsf{echo}(R))\ \textbf{(Inv)}\ \text{and}\ R \neq \mathsf{nil}\ \}\}$$

$$\{\{\ \mathsf{echo}(\mathsf{this.vals}) = \mathsf{concat}(\mathsf{rev}(\mathsf{cons}(R.\mathsf{hd}, \mathsf{cons}(R.\mathsf{hd}, S))), \mathsf{echo}(R.\mathsf{tl}))\ \}\}$$

Problem 3

Fill in the body of the (6) unit tests for the `values` method from problem 2. Include comments explaining the test cases

```
// ____________________________________________________________
assert.deepStrictEqual(
    makeEchoCursor(________________________________________).values(),
    ___________________________________________________________);
```

```
values = (): List<number> => {
  let R: List<number> = this.vals;
  let S: List<number> = nil;
  {{ Inv: echo(this.vals) = concat(rev(S), echo(R)) }}
  while (R !== nil) {
    S = cons(R.hd, cons(R.hd, S));
    R = R.tl;
  }
  {{ echo(this.vals) = rev(S) }}
  return rev(S);
};
```

Problem 4a

Fill in the body of the “`valueAt`” method of `EchoCursor` so it satisfies the spec.

Your code may *not* call any functions other than “at” from Problem 1

```
// @returns at(n, obj)
valueAt = (n: number): number => {




};
```

```
class EchoCursor implements IntCursor {
  // AF: obj = echo(this.vals)
  readonly vals: List<number>;  // list before echo-ing

  constructor(vals: List<number>) {
    this.vals = vals;
  }
```

Problem 4b

Explain, in English, why your implementation is correct.

To get full credit, your explanation must reference the AF (since the spec is in terms of the abstract state, obj) and the two facts noted in Problem 1.

```
// @returns at(n, obj)
valueAt = (n: number): number => {
  const m = Math.floor(n / 2);
  return at(m, this.vals);
};
```

```
// AF: obj = echo(this.vals)
readonly vals: List<number>;  // list before echo-ing
```

- **Fact A**: at$(2m, \text{echo}(S)) = $ at(m, S) for any $m : \mathbb{N}$ and S : List.
- **Fact B**: at$(2m + 1, \text{echo}(S)) = $ at(m, S) for any $m : \mathbb{N}$ and S : List.