CSE 331
Software Design & Implementation

Fall 2023

Section 2 – HW2: Correctness, Specifications, & Testing

Administrivia

- HW2 released tonight, due next Wednesday **11pm**
  - No more than **one** late day per assignment
  - 4 late days in total
- Section solutions posted tonight & recording later on

Reminders:

- Check out ed guidelines on ed!
- Can attend any OH! Queue posted on ed, most in-person TAs will just use whiteboard

Where are we?

- ✓ Basics of Typescript
- ✓ Basics of the browser

- Math notation & specifications
- Correctness concepts
- Testing

- ○ Reasoning techniques (for levels 1-3)
- ○ Abstraction
- ○ App design / more complex React apps
- ○ + more!

Review – Correctness

Level	Description	Testing	Tools	Reasoning
-1	small # of inputs	exhaustive		
0	straight from spec	heuristics	type checking	code reviews
1	no mutation	“	libraries	calculation induction
2	local variable mutation	“	“	Floyd logic
3	array / object mutation	“	“	rep invariants

- Test all possible cases if reasonable, then use heuristics to approximate
- As code increases ⬆️ in complexity, formality and complexity of reasoning technique must increase ⬆️ too
- 3 is “worst case” for how difficult it is to be confident it’s correct

Question 1

(a) Consider the following mathematical function defined on the integers 1, 2, 3, and 4:

$$\begin{aligned} \textbf{func}\ f(1) &:= 2 \\ f(2) &:= 3 \\ f(3) &:= 4 \\ f(4) &:= 1 \end{aligned}$$

If we implement this directly in TypeScript using a `switch` statement, what level of correctness is required?

Question 1

(b) Consider the following mathematical function defined on the inputs n and b, where n is 1, 2, 3, or 4 and b is true or false. It is defined in terms of the function f defined in part (a).

$$\begin{aligned} \textbf{func}\ g(n, \mathsf{T}) &:= f(n) \\ g(n, \mathsf{F}) &:= f(n) \end{aligned}$$

If we implement this in TypeScript using an `if` statement (on b), what level of correctness is required?

Question 1

(c) Consider the following mathematical function defined on the inputs n and x, where n is 1, 2, 3, or 4 and x is any integer. It is defined in terms of the function f defined in part (a).

$$\textbf{func } h(n, x) := f(n) + x$$

If we implement this in TypeScript using a single `return` statement, what level of correctness is required?

Question 1

(d) Suppose that we implement the function h with the following TypeScript code. It calls f, which we will assume is implemented in TypeScript with one conditional.

```
const h = (n: number, x: number): number => {
  let y = f(n);
  while (x > 0) {
    y = y + 1;
    x = x - 1;
  }
  return y;
}
```

What level of correctness is required now?

Question 2 – Set up

Clone the starter code:

```
git clone https://gitlab.cs.washington.edu/
      cse331-23au-materials/sec-levels.git
```

Then run:

```
npm install --no-audit
```

In the `sec-levels` directory

Try:

```
npm run test
```

to confirm that all the test fail at this point

Question 2 – Preface

- This question asks us to implement functions incorrectly
- Tests verify that *some* cases produce correct output, and that particular cases produce incorrect output
- Completing the problem correctly = the tests pass = the function is incorrect

- **Why this exercise?**
  - Tests give confidence that implementations are correct
  - typos, misplacing boundaries, forgetting cases are realistic mistakes to make
  - With the *wrong* set of tests or *not enough* tests, mistakes can go unnoticed, giving false confidence in correctness
  - So, we use heuristics!!

Question 2

(a) Fill in the code for the function `quadratic1` and `quadratic2` in `src/funcs.ts` so that it passes the tests provided in `src/funcs_test.ts` but is wrong (not correct on all inputs).

(b) Fill in the code for the function `abs_value` in `src/funcs.ts` so that it passes the tests provided in `funcs_test.ts` but is wrong.

Your implementation must be a single "`if`" statement (i.e., a conditional), with one branch returning "`x`" and the other branch returning "`-x`". You can choose the branch condition.

Run tests with: **`npm run test`**

Question 5a

(a) We included 4 tests for `abs_value`, two for each branch. Why was that not enough to detect the problem? What heuristic did we forget about?

Question 2

(c) If our code does pass all the tests required by our heuristics, does that *guarantee* that it is correct?

Review – Math Notation

$\mathbb{N}$	all non-negative integers ("natural" numbers)
$\mathbb{Z}$	all integers
$\mathbb{R}$	all real numbers
$\mathbb{B}$	the boolean values (T and F)
$\mathbb{S}$	any character
$\mathbb{S}^*$	any sequence of characters ("strings")

Made up by Kevin :)

- **Union**: $A \cup B$ set including everything in A *and* B
- **Tuple**: $A \times B$ all pairs (a, b) where $\mathrm{a} \in A$ and $\mathrm{b} \in B$
- **Record**: $\{x: A, y: B\}$ all records with fields x, y of types A, B

Review – Math Notation

- **Pattern matching**: defining function based on input cases
  - Exactly **one** rule for every valid input

  ex: $\text{func } f(0) \coloneqq 0$

  $f(\text{n+1}) \coloneqq n \qquad \text{for any n: } \mathbb{N}$

  → "n+1" is signifying that the input must be > 0 since the smallest value $\text{n: } \mathbb{N}$ would be 0

- **Side conditions**: limiting/specifying input in the right column, cleans things up, pattern matching preferred

- See the course website > Calendar > 10/4 lecture notes: "Math Notation" for more!

Question 3

$$
\begin{aligned}
\textbf{func}\ \text{half}(\text{null}) &:= 0 \\
\text{half}(\text{undefined}) &:= 0 \\
\text{half}(n : \mathbb{N}) &:= n/2 && \text{if } n \text{ is even} \\
\text{half}(n : \mathbb{N}) &:= -(n+1)/2 && \text{if } n \text{ is odd}
\end{aligned}
$$

(a) What is the type for the function half? (There are 2 possibilities.) Use the notation half : $A \to B$ to indicate that half takes inputs of type A and produces outputs of type B.

Question 3

$\text{half} : (\text{null} \mid \text{undefined} \mid \mathbb{N}) \to \mathbb{Z}$ or $\text{half} : (\text{null} \mid \text{undefined} \mid \mathbb{N}) \to \mathbb{R}$.

(b) What would the declarations of this function look like in TypeScript based on the type?

(c) Implement the mathematical function half as a Typescript function in `funcs.ts`. Make sure it is exported.

Run tests with: `npm run test`

Question 4

```
const maybeDouble = (t: {b: boolean, v: [boolean, number]}): number => {
  if (t.b) {
    if (t.v[0]) {
      return 2 * t.v[1];
    } else {
      return t.v[1];
    }
  } else {
    return 0;
  }
};
```

How would you translate this into our math notation using pattern matching?