CSE 331: Software Design & Engineering Fall 2024

Homework 6 Code

Due: Monday, November 11th, 11pm

If you have not completed HW6 written. Close this file and finish the written component before starting the coding parts in this worksheet.

To get started, check out the starter code for this assignment:

```
git clone https://gitlab.cs.washington.edu/cse331-24au-materials/hw6-digits.git
```

Navigate to the `hw6-digits` directory and run `npm install --no-audit`. Run tests with the command `npm run test` and run the linter with the command `npm run lint`.

Starting with this homework, mutation is no longer prohibited (provided you use it correctly)! If you have the comfy-tslint extension in VS Code, you will need to update a setting to allow mutation and prevent it from giving you errors when you do things like use `let` and reassign variables:

Open the **comfy-tslint extension** and press the **gear icon** to the right of the "Disable" and "Uninstall" buttons. Open **"Extension Settings"** from the drop down options that appear. The check the box to enable the **"Comfy TS Linter: Allow Mutation"** setting, and save your settings.

If you do not have the extension installed you will not need to update anything. You can continue to use the `npm run lint` command which we will update for these coming assignments to permit mutation.

Submission

After completing all tasks to follow, submit your solutions on Gradescope. The following completed files should be submitted to **"HW6 Code"**:

`int_sqrt.ts` `int_sqrt_test.ts` `even_non_zeros.ts` `even_non_zeros_test.ts`

Wait after submitting to make sure the autograder passes, and leave yourself time to resubmit if there's an issue. The autograder will run your tests, additional staff tests, and the linter.

Task 7 – The Best Things In Life Are Three [10 pts]

In this problem, we will translate your written code for functions into TypeScript code.

These translations should be a direct translation from your written work.

Just like last week, we have not provided any tests for these functions. (You will write those in the next part!) However, you proved some assertions related to these functions in your written assignment, so if you translate the code directly, you should already have some confidence that it is correct.

(a) Translate your written code from HW6 Written Task 3, into TypeScript code in `int_sqrt.ts`.

Complete the TODOs in the body of the function by uncommenting the code and filling in the blank lines with the code from your written work.

If you have realized that you made mistakes in your original written implementation, feel free to fix those when you type them up here to get the tests to pass.

(b) Now, translate your written code from Task 5, into TypeScript code in `even_non_zeros.ts`.

Complete the TODOs in the body of the function by uncommenting the code and filling in the blank lines with the code from your written work.

If you have realized that you made mistakes in your original written implementation, feel free to fix those when you type them up here to get the tests to pass.

Look at specs provided for these functions. They describe the behavior in math definitions and English descriptions (matching the written instructions). These functions are implemented with a loop instead of "straight from the spec", so it is important that we know the code implements the behavior the users expect, given the spec, which is why we practice formally proving that correctness in the written part.

Below is another example for a TypeScript implementation of "sum" that uses a loop but still implements the spec:

```
/**
 * Finds the sum of all elements in the list
 * @param L list of ints to sum elements of
 * @returns sum(L) where sum is defined as follows:
 *    sum: List<Z> -> Z
 *    sum(nil)  := nil
 *    sum(a::L) := a + sum(L)
 */
const sum = (L: List<bigint>) => {
   let x: bigint = 0n;
   let ys: List<bigint> = L;
   // Inv: sum(L) = sum(ys) + x
   while (ys !== nil) {
       x = x + ys.hd;
       ys = ys.tl;
   }
   // {{ sum(L) = x }}
   return x;
}
```

Task 8 – Wicked Witch of The Test [10 pts]

Now, for our final step in gaining confidence our code is correct, it's time to test the functions we translated in the last part!

Your tests should follow the testing requirements for this course (see the notes on testing for a reminder). Additionally, write short labels describing which coverage requirement is met by each test. See the example from HW4 for reference.

(a) Write tests for int_sqrt in `int_sqrt_test.ts`.

(b) Write tests for even_non_zeros in `even_non_zeros_test.ts`.