CSE 331: Software Design & Implementation Winter 2026

# Homework 2

Due: Wednesday, January 21st, 11:59pm

## Task 1 – The Test Man Speech [10 pts]

Complete these tasks or each of the following functions.

- State the number of tests required to meet the 331 coverage requirements and explain why that is the required number.
- Then, describe a specific set of tests to use. The size of set should match the number of tests you stated before. Describe each test by giving the input (identify a specific input rather than saying, e.g., "some positive number") and briefly state what portion of the function it tests (e.g., "first if statement coverage", "for 0 iterations").

**a)** The function $j : (\text{List}) \to \text{List}$ defined by

$$\begin{aligned} j(\text{nil}) &= \text{nil} \\ j(a :: \text{nil}) &= \text{nil} \\ j(a :: b :: L) &= b :: a :: j(L) \end{aligned}$$

**b)**

```
/** @requires b >= 1 and n >= 0 */
public static int pow(int b, int n) {
  if (n == 0) {
    return 1;
  }
  return b * pow(b, n - 1);
}
```

## Task 2 – The Minner Takes It All [14 pts]

The functions max, min : $(\mathbb{Z}, \mathbb{Z}) \to \mathbb{Z}$ are defined as follows:

$$\begin{aligned} \mathsf{max}(a, b) &:= a \quad \text{if } a \geqslant b \\ \mathsf{max}(a, b) &:= b \quad \text{if } a < b \end{aligned}$$

$$\begin{aligned} \mathsf{min}(a, b) &:= b \quad \text{if } a \geqslant b \\ \mathsf{min}(a, b) &:= a \quad \text{if } a < b \end{aligned}$$

**a)** Using these definitions, prove the following claim **by cases**:

$$\mathsf{min}(a, b) \leqslant \mathsf{max}(a, b)$$

For this problem, when you cite the definition of min or max in your proof, also mention which line of the definition you are citing.

**b)** Now consider the code implementations of max and min.

Note that the names of the code functions match the names of the math functions. The comments describe which math function each code function is implementing.

```
// returns max(a, b), where max is the math function defined earlier
public static int max(int a, int b) {
  if (a >= b) {
    return a;
  } else {
    return b;
  }
}

// returns min(a, b), where min is the math function defined earlier
public static int min(int a, int b) {
  if (a >= b) {
    return b;
  } else {
    return a;
  }
}
```

State the number of tests needed to meet our coverage requirements. List a set of test inputs (of the form $a = ...$ and $b = ...$) that achieves this for both functions, an explain which parts of the code each inputs would cover. You may use the same set of test inputs for both functions.

## Task 3 – A Tail of Two Cities [16 pts]

The functions head : (List) $\to \mathbb{Z}$ and tail : (List) $\to$ List are defined only on non-nil lists, with the following definitions:

$$\text{head}(x :: L) = x$$
$$\text{tail}(x :: L) = L$$

Now consider the function swap : (List) $\to$ List, defined as follows:

$$\text{swap}(\text{nil}) = \text{nil}$$
$$\text{swap}(x :: \text{nil}) = x :: \text{nil}$$
$$\text{swap}(x :: y :: L) = y :: x :: \text{swap}(L)$$

You will use these functions in the problem below.

Let $x$ and $y$ be integers and $L$ a list. Complete each of the following proofs **by calculation**.

Include an explanation on each step where a given fact is used. You may skip an explanation only if the claim written in that step is *literally* a known fact. It is also fine to skip an explanation for simple algebraic equivalences (e.g., 3 - 0 = 3).

Also include an explanation on each step where a definition is used. You do not need to include an explanation on lines that simply rewrite an expression in alternative but equivalent notation. For example, no explanation is needed when replacing "$[x]$" by "$x ::$ nil", which means the same thing. For this problem, you do *not* need to say which line of a definition you are citing; you can just say which function's definition you are citing.

**a)** $\text{head}(\text{nil} +\!\!+ [x]) = x$.

**b)** $\text{tail}(\text{nil} +\!\!+ [x]) = \text{nil}$.

**c)** $\text{head}((y :: L) +\!\!+ [x]) = y$.

**d)** $\text{tail}(\text{swap}(x :: y :: \text{nil})) = x :: \text{nil}$

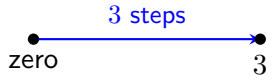
**e)** $\text{head}(\text{tail}(\text{swap}([x] +\!\!+ y :: z :: L))) = x$

## Task 4 – Succs to Succ [15 pts]

Recall the definition of the natural numbers $\mathbb{N}$:

$$\textbf{type } \mathbb{N} = \text{zero} \mid \text{succ}(\mathbb{N})$$

We can think of each natural number as counting out steps along the $x$ axis. A single call to succ is 1 step. We take multiple steps by making nested calls to succ. For example, the natural number succ(succ(succ(zero))) is taking three steps along the $x$ axis starting from zero, so it represents the number 3.




We can then define addition of natural numbers, add : $(\mathbb{N}, \mathbb{N}) \to \mathbb{N}$, by thinking of add$(n, k)$ as taking $k$ steps and then $n$ steps along the $x$ axis. For example, add$(3, 5)$, adding 3 to 5 looks like this:


We can accomplish this with the following definition.

$$\begin{aligned} \text{add}(\text{zero}, k) &= k \\ \text{add}(\text{succ}(n), k) &= \text{succ}(\text{add}(n, k)) \end{aligned}$$

The first line of the definition says that $k$ steps and then zero steps is just $k$ steps. The second line says that taking $k$ steps and then $n + 1$ steps is the same as taking $k$ steps and then $n$ steps and then one more step. The resulting definition of add is structurally recursive on its first parameter.

Prove that add$(n, \text{succ}(k)) =$ add$(\text{succ}(n), k)$ holds for all $n$ and $k$ by **structural induction** on $n$. For this problem only, use the notation zero and succ in your proof; do not use the notations "0" or "+1".

## Task 5 – Extra Credit: If You Catch My Shift [0 pts]

The functions shift-left, shift-right : (List) $\to$ List rotate the elements in a list forward by one index and backward by one index, respectively, wrapping around at the beginning and end. They are defined formally as follows:

$$\text{shift-left}(\text{nil}) = \text{nil}$$
$$\text{shift-left}(x :: L) = L +\!\!+ [x]$$

$$\text{shift-right}(\text{nil}) = \text{nil}$$
$$\text{shift-right}(x :: L) = \text{last}(x :: L) :: \text{init}(x :: L)$$

You will use these functions in the problem below.

**a)** Let $x$ be an integer. Prove $\text{last}(L +\!\!+ [x]) = x$ by structural induction on $L$.

**b)** Let $x$ be an integer. Prove $\text{init}(L +\!\!+ [x]) = L$ by structural induction on $L$.

**c)** Prove that $\text{shift-right}(\text{shift-left}(L)) = L$ holds by cases.

## Task 6 – Optional: Be Add That It Happened [0 pts]

Recall the definitions of $\mathbb{N}$ and add : $(\mathbb{N}, \mathbb{N}) \to \mathbb{N}$ from Task 4.

**a)** Prove that $\text{add}(n, \text{zero}) = n$ by structural induction.

**b)** Prove that $\text{add}(n, m) = \text{add}(m, n)$ holds for all $n$ and $m$ by structural induction on $n$.