Quiz Section 2: Proofs and Testing – Solutions

Task 1 – Calc You Later [12 pts]

Let a, b, c, d, e be integers. Complete each of the following proofs **by calculation**.

Include a justification on every step where a given fact is used. You can skip such an explanation only if the claim written in that step is itself, *literally* a known fact. It is also fine to cite a fact that is equivalent (via simple algebra) to a known fact. You do **not** need to cite algebra.

a) Given that $a = 1$ and $b = 1$, it follows that $a = 2b - 1$.

$$\begin{aligned} a &= 1 \\ &= 2 - 1 \\ &= 2 \cdot 1 - 1 \\ &= 2b - 1 && \text{since } b = 1 \end{aligned}$$

b) Given that $a = 1$, $b = 2a - 1$, and $c > 0$, it follows that $(b-1)^2 < c$.

$$\begin{aligned} (b-1)^2 &= (2a - 1 - 1)^2 && \text{since } b = 2a - 1 \\ &= (2a - 2)^2 \\ &= (2 \cdot 1 - 2)^2 && \text{since } a = 1 \\ &= 0^2 \\ &= 0 \\ &< c && \text{since } c > 0 \end{aligned}$$

c) Given that $d = b + 1$, $c = a - 8$, and $e = a + 8b$, it follows that $e = c + 8d$.

$$\begin{aligned} e &= a + 8b \\ &= c + 8 + 8b && \text{since } a = c + 8 \\ &= c + 8 + 8(d - 1) && \text{since } b = d - 1 \\ &= c + 8 + 8d - 8 \\ &= c + 8d \end{aligned}$$

d) Given that $b = 2a - 1$, $d = a^2$, and $d + b + 2 < c$, it follows that $(a+1)^2 < c$.

$$\begin{aligned} (a+1)^2 &= a^2 + 2a + 1 \\ &= a^2 + 2a - 1 + 2 \\ &= a^2 + b + 2 && \text{since } b = 2a - 1 \\ &= d + b + 2 && \text{since } d = a^2 \\ &< c && \text{since } d + b + 2 < c \end{aligned}$$

Task 2 – Absolutely Positive [6 pts]

Let x be an integer and L a list. Complete the following proof **by cases**.

Your individual calculations should include explanations as in the previous problems. When citing a function definition that uses a side condition, your explanation must not only say which function's definition is being used but also what side condition is known to hold so that the reader can see what line of the definition you are referring to.

Let abs : $\mathbb{Z} \to \mathbb{Z}$ be defined as follows:

$$\begin{aligned} \mathsf{abs}(x) &= -x && \textbf{if } x < 0 \\ \mathsf{abs}(x) &= x && \textbf{if } x \geqslant 0 \end{aligned}$$

Prove that $\mathsf{abs}(\mathsf{abs}(x)) = \mathsf{abs}(x)$.

Suppose that $x < 0$. Then, we can see that

$$\begin{aligned} \mathsf{abs}(\mathsf{abs}(x)) &= \mathsf{abs}(-x) && \text{def of abs (since } x < 0) \\ &= -x && \text{def of abs (since } -x > 0) \\ &= \mathsf{abs}(x) && \text{def of abs (since } x < 0) \end{aligned}$$

Now, suppose that $x \geqslant 0$. Then we can see that

$$\mathsf{abs}(\mathsf{abs}(x)) = \mathsf{abs}(x) \quad \text{def of abs (since } x \geqslant 0)$$

These two cases are exhaustive, so we have proven the claim holds in general.

Task 3 – Keeping It Cool [8 pts]

The functions keep, skip : (List) → List remove half the elements in a list. These two functions keep and skip every other element of the passed in list. The keep function includes the first element but skips the one after it, while skip drops the first element but keeps the one after that. They are defined formally as follows:

$$\begin{aligned} \mathsf{keep}(\mathsf{nil}) &= \mathsf{nil} \\ \mathsf{keep}(x :: L) &= x :: \mathsf{skip}(L) \end{aligned}$$

$$\begin{aligned} \mathsf{skip}(\mathsf{nil}) &= \mathsf{nil} \\ \mathsf{skip}(x :: L) &= \mathsf{keep}(L) \end{aligned}$$

Also, recall the function echo : (List) → List, which was defined in class as follows:

$$\begin{aligned} \mathsf{echo}(\mathsf{nil}) &= \mathsf{nil} \\ \mathsf{echo}(x :: L) &= x :: x :: \mathsf{echo}(L) \end{aligned}$$

and the function sum : (List) → $\mathbb{N}$, which was defined in class as follows:

$$\begin{aligned} \mathsf{sum}(\mathsf{nil}) &= 0 \\ \mathsf{sum}(x :: L) &= x + \mathsf{sum}(L) \end{aligned}$$

You will use these functions in the problem below.

Prove that $\mathsf{sum}(\mathsf{skip}(\mathsf{echo}(L))) = \mathsf{sum}(L)$ holds by structural induction on L.

Define $P(L)$ to be the claim $\mathsf{sum}(\mathsf{skip}(\mathsf{echo}(L))) = \mathsf{sum}(L)$. We will prove that this holds for all values of L by structural induction.

Base Case. We can see that $P(\mathsf{nil})$ holds as follows:

$$\begin{aligned} \mathsf{sum}(\mathsf{skip}(\mathsf{echo}(\mathsf{nil}))) &= \mathsf{sum}(\mathsf{skip}(\mathsf{nil})) && \text{def of echo} \\ &= \mathsf{sum}(\mathsf{nil}) && \text{def of skip} \end{aligned}$$

Inductive Hypothesis. Suppose that $P(L)$ holds for some list of integers L.

Inductive Step. Let x be an arbitrary integer. We can see that $P(x :: L)$ holds as follows:

$$\begin{aligned} \mathsf{sum}(\mathsf{skip}(\mathsf{echo}(x :: L))) &= \mathsf{sum}(\mathsf{skip}(x :: x :: \mathsf{echo}(L))) && \text{def of echo} \\ &= \mathsf{sum}(\mathsf{keep}(x :: \mathsf{echo}(L))) && \text{def of skip} \\ &= \mathsf{sum}(x :: \mathsf{skip}(\mathsf{echo}(L))) && \text{def of keep} \\ &= x + \mathsf{sum}(\mathsf{skip}(\mathsf{echo}(L))) && \text{def of sum} \\ &= x + \mathsf{sum}(L) && \text{Ind. Hyp.} \\ &= \mathsf{sum}(x :: L) && \text{def of sum} \end{aligned}$$

Conclusion. $P(L)$ holds for all lists of integers L by structural induction.

Task 4 – The Test-Laid Plans [12 pts]

For each of the following functions, state the number of tests required to meet our coverage requirements and explain why that is the required number.

Then, describe a specific set of tests to use (with the same number of tests you as said before). Describe each test by giving the input (identify a specific input rather than saying, e.g., "some positive number"), stating which portion of the function it tests, and explaining why our rules require that test.

a)

```
public static int f(int n) {
  if (n < 0) {
    return -2 * n;
  } else {
    return 3 * n;
  }
}
```

This function requires 2 tests to achieve statement coverage, one with $n < 0$ and one with $n \geqslant 0$. Those values will also give us branch coverage. Loop coverage holds vacuously.

Particular tests would be $f(-1) = 2$ for the top branch and $f(2) = 4$ for the bottom.

b)

```
public static int h(int n) {
  if (n <= 0) {
    return 1;
  } else {
    return 2 + h(n / 3);
  }
}
```

This function requires 3 tests to achieve loop coverage. The same three tests can also provide statement and branch coverage.

Valid tests would be $h(0) = 1$ for 0 iterations, $h(1) = 3$ for 1 iteration, and $h(3) = 5$ for 2 iterations.