CSE 331: Software Design & Implementation

Section 3

The problems that follow make use of the following inductive type, representing lists of integers

$$\textbf{type}\ \text{List} := \quad \text{nil} \quad | \quad \text{cons}(\text{hd} : \mathbb{Z},\ \text{tl} : \text{List})$$

In lecture, we saw some standard functions on list. One was len, which returns the length of the list. It is defined formally using recursion as follows:

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

In the next homework, we will also use the function sum, which returns the sum of the integers in the list:

$$\begin{aligned} \textbf{func}\ \text{sum}(\text{nil}) &:= 0 \\ \text{sum}(\text{cons}(a, L)) &:= a + \text{sum}(L) \quad \text{for any } a : \mathbb{Z} \text{ and } L : \text{List} \end{aligned}$$

1. Sugar and Spice and Everything Twice

We are asked to write a function "twice" that takes a list as an argument and "returns a list of the same length but with every number in the list multiplied by 2".

(a) This is an English definition of the problem, so our first step is to formalize it. Let's start by writing this out in more detail. Fill in the blanks showing the result of applying twice to lists of different lengths.

nil	_____
cons(a, nil)	_____
cons$(a, \text{cons}(b, \text{nil}))$	_____
cons$(a, \text{cons}(b, \text{cons}(c, \text{nil})))$	_____
...	

(b) The previous list of examples is not a formal definition. It does not tell us, for example, what twice does to a list of length 4. More generally, any time we see "...", the definition is probably not formal.

Write a formal definition of twice using recursion.

(c) If we translated this into TypeScript code in the most direct manner (level 0), what heuristic should we use to get a set of subdomains? What specific tests should we use to make sure that everything is correct?

2. Twice Things Up

You see the following snippet in some TypeScript code. It uses `cons` and `nil`, which are TypeScript implementations of "cons" and "nil", and also `equal`, which is a TypeScript implementation of "=" on lists.

```
if (equal(L, cons(1, cons(2, nil)))) {
  const R = cons(2, cons(4, nil));  // = twice(L)
  return cons(0, R);                // = twice(cons(0, L))
}
```

The comments show the definition of what *should* be returned (the specification), but the code is *not* a direct translation of those. Below, we will use reasoning to prove that the code is correct.

(a) Using the fact that $L = \text{cons}(1, \text{cons}(2, \text{nil}))$, prove by calculation that $\text{twice}(L) = R$, where R is the constant list defined in the code. I.e., prove that

$$\text{twice}(L) = \text{cons}(2, \text{cons}(4, \text{nil}))$$

(b) Using the facts that $L = \text{cons}(1, \text{cons}(2, \text{nil}))$ and $R = \text{cons}(2, \text{cons}(4, \text{nil}))$, prove by calculation that the code above returns the correct value, i.e., prove that

$$\text{twice}(\text{cons}(0, L)) = \text{cons}(0, R)$$

Feel free to cite part (a) in your calculation.

3. Miami Twice

We are asked to write a function that takes a list as an argument and "returns a list of the same length but with *every other* number in the list, *starting with the first number*, multiplied by 2".

The first number in the list is at index 0, which is even; the second number in the list is at index 1, which is odd; the third number in the list is at index 2, which is even; and so on. Hence, we will call this function twice-evens because it multiples the numbers at even indexes by two and leaves those at odd indexes unchanged.

(a) The definition of the problem was in English, so our first step is to formalize it. Let's start by writing this out in more detail. Fill in the blanks showing the result of applying twice-even to lists of different lengths.

nil	_____
cons$(a,$ nil$)$	_____
cons$(a,$ cons$(b,$ nil$))$	_____
cons$(a,$ cons$(b,$ cons$(c,$ nil$)))$	_____
...	

(b) The previous list of examples is not a formal definition (because of the "...").

Write a formal definition of this function, twice-evens, using recursion. In order to do so, you may need to define more than one function!

(c) If we translated this into TypeScript code in the most direct manner (level 0), what tests (if any) should we include to make sure that everything is correct?

4. It's Raining Len

You see the following snippet in some TypeScript code. It uses `twice_evens`, which is a TypeScript implementation of twice-evens from the previous problem, as well as `len` from before.

```
return 2 + len(twice_evens(L));  // = len(twice-evens(cons(3, cons(4, L))))
```

The comment shows the definition of what should be returned (the specification), but the code is not a direct translation of that. Below, we will use reasoning to prove that the code is correct.

(a) Let a and b be any integers. Prove by calculation that

$$\text{len}(\text{twice-evens}(\text{cons}(a, \text{cons}(b, L)))) = 2 + \text{len}(\text{twice-evens}(L))$$

This form of argument is called a direct proof: a style of argument where we use variables to stand in for values (a and b for integer values, in this case) and prove that the claim holds regardless of the actual value of that variable. Once the proof is complete, we are allowed to substitute in any concrete values for the variables and know that the resulting fact is true. It must be true because substituting those values into the calculation would give us a proof for those specific values!

(b) Explain why the direct proof from part (a) shows that the code is correct according to the specification (written in the comment).