

# CSE 331: Software Design & Implementation

---

## Section 4

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

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

Below, we will also use the function sum, which returns the sum of the integers in the list:

**func** sum(nil) := 0  
sum(cons(a, L)) := a + sum(L) for any  $a : \mathbb{Z}$  and  $L : \text{List}$

the function twice, which doubles each number in the list:

**func** twice(nil) := nil  
twice(cons(a, L)) := cons(2a, twice(L)) for any  $a : \mathbb{Z}$  and  $L : \text{List}$

the functions twice-evens and twice-odds, which double the numbers at even and odd indexes in the list:

**func** twice-evens(nil) := nil  
twice-evens(cons(a, L)) := cons(2a, twice-odds(L)) for any  $a : \mathbb{Z}$  and  $L : \text{List}$   
**func** twice-odds(nil) := nil  
twice-odds(cons(a, L)) := cons(a, twice-evens(L)) for any  $a : \mathbb{Z}$  and  $L : \text{List}$

and the function swap, which swaps adjacent numbers in the list:

**func** swap(nil) := nil  
swap(cons(a, nil)) := cons(a, nil) for any  $a : \mathbb{Z}$   
swap(cons(a, cons(b, L))) := cons(b, cons(a, swap(L))) for any  $a, b : \mathbb{Z}$  and  $L : \text{List}$

## 1. Any Sort in a Storm

We are asked to write a function pseudo-sort that takes a list of numbers as an argument, “looks at the first two numbers in the list, orders the pair to place the smaller of the two in the front, and then continues through the following pairs in the list after the first element”.

For example: if the first two elements in a list are 5, 1 they would be reordered so the 1 comes first: 1, 5, and the next pair to look at would have the 5 as the first element.

- (a) This is an English definition of the problem, so our first step is to formalize it.

Write a formal definition of pseudo-sort using recursion.

Note that, in order to do this, you will need to include extra conditions on the patterns used in the definition. You can write extra conditions in the third column (where we usually write, e.g., “for any  $a : \mathbb{Z} \dots$ ”) by writing “with” or “if” and then the condition.

Also, if the conditions do not fit on one line, you can replace them with a label, e.g., “A”, and then define that label on another line by writing “where  $A$  is  $\dots$ ”. (Likewise, if any expression is too long, you can replace part of the expression by a new variable, e.g., “ $n$ ”, and then define that variable on another line by writing “where  $n := \dots$ ”.)

- (b) Show by example that pseudo-sort does not actually sort the list.

Note that, when you are doing a calculation and applying a definition that has an extra condition, you should point out *why* that condition holds in your explanation in the right column.

## 2. Here Comes the Sum

You see following snippet in some TypeScript code:

```
const s = sum(L);  
...  
return 2 * s; // = sum(twice(L))
```

This code claims to calculate the answer  $\text{sum}(\text{twice}(L))$ , but it actually returns  $2 \text{sum}(L)$ . Prove this code is correct by showing that  $\text{sum}(\text{twice}(S)) = 2 \text{sum}(S)$  holds for any list  $S$  by structural induction.

### 3. Can You Sum a Few Bars?

In HW3, you proved the following holds when  $L$  has length 3:

$$\text{sum}(\text{twice-evens}(L)) + \text{sum}(\text{twice-odds}(L)) = 3 \text{sum}(L)$$

Prove that this holds for any list  $S$  by structural induction.

#### 4. Swapaholic

Prove by cases that  $\text{swap}(\text{cons}(a, L)) \neq \text{nil}$  for any integer  $a : \mathbb{Z}$  and list  $L$ .