

# Homework # 7

## due October 21

### 1 Reading

Please read Chapter 11 in your textbook.

### 2 Problems

Please do the following problems

1. Enums can be seen as a variation on “variants”. Write new typing rules (on paper, not SASyLF) for a type `T` written like `[red,green,blue]` with new syntax:

$$\begin{array}{l}
 t ::= \dots \\
 \quad | \quad n_i \text{ as } [n_1, \dots, n_m] \quad (\text{a new kind of value}), \\
 \quad | \quad \text{case } t \text{ of } n_1 \rightarrow t_1 \mid \dots \mid n_m \rightarrow t_m.
 \end{array}$$

2. Use `fullsimple` (using `fix`, *NOT* `letrec`) to implement infinite lists similarly to tables (see page 137). An infinite list will map a natural number to a natural number. In particular, we do not need options. Implement the following functions:
  - `nats`: a list of all natural numbers, in order starting from zero.
  - `cons`: to add a new element on the front of the list.
  - `tail`: to remove the front element from the list.
  - `head`: to return the first element of the list.
  - `sumN`: to add the first “n” elements of a list (this function takes two parameters).
  - `filter`: taking a predicate (function of type `Nat -> Bool`) and a list, return a list with all elements that are true for the predicate, in their original order.

In particular:

```
sumN 3 (filter iseven (cons 7 nats))
```

should evaluate to 6. Your code should run correctly; you will need to write definitions of `plus` (obvious) and `iseven` (in book) as well. Don’t worry about efficiency.

Leave your whole program in `list.f` in your AFS volume for Homework #6.

### 3 Proofs

Modify the proofs of type soundness of the simply-typed lambda calculus with “unit” types to add sums (See Figure 11-9). A skeleton file is available which gives the syntax (slightly modified from that in the textbook to avoid the keyword `case` and reserved operator `|`). You will need to define evaluation and type rules. You should follow the book as closely as possible.