Quiz 2

Problem 1. The Scheme special form **or** behaves as follows: if ev-val(A) = `#f`, then ev-val(`(or` A B`)`) = ev-val(B). If ev-val(A) is not `#f`, then ev-val(`(or` A B`)`) = ev-val(A). (We restrict `or` to having exactly *two* components, viz., no `(or` A B C`)`.)

1(a) Give rewrite rules which completely determine the behavior of `or` according to the above specification.

1(b) Give a simple example expression, A, which shows why `(or` A `'dummy)` cannot be regarded as sugar for `(if` A A `'dummy)`.

1(c) Show that, nevertheless, `or` can be regarded as sugar.

1(d) Modify the definition of *control context* (Handout 14, p. 6) to handle the `or` special form, and carefully define an `or`-evaluation redex (cf. Handout 14, p. 6).

Problem 2. Let $\mathcal{H}_0$ ::= $\{\, M \mid M$ is a Scheme expression and `(`M `‘`M`)` halts $\}$.

2(a) Explain why Scott's Rice's Theorems do *not* directly imply that $\mathcal{H}_0$ is undecidable.

2(b) Prove that $\overline{\mathcal{H}_0}$ is not half-decidable (that is, not r.e.).

2(c) Explain why $\mathcal{H}_0$ is half-decidable.

Problem 3. Let $\mathcal{H}_0$ be as in Problem 2, and let

$$\mathcal{N} ::= \{\, \texttt{(0}\ S\texttt{)} \mid S \in \mathcal{H}_0 \} \cup \{\, \texttt{(1}\ S\texttt{)} \mid S \notin \mathcal{H}_0 \}$$

where S ranges over S-expressions.

3(a) Suppose N was a Scheme expression which was a half-decider for $\mathcal{N}$. Fill in the expressions ①, ② below so that the following expression is a decider for $\mathcal{H}_0$:

```
(lambda (qs)
    (define (try n)
        (cond ( (<step-smeval> (list 'N ①) n) #t)
              ( (<step-smeval>      ②      n) #f)
              (else (try (+ n 1)))))
    (try 0))
```

3(b) Explain why neither $\mathcal{N}$ nor $\overline{\mathcal{N}}$ is half-decidable.

Problem 4. We consider Scheme extended with a $\langle$primitive procedure id$\rangle$, `int->sym`, which computes a one-to-one total function

$$\text{int-to-sym} : \mathbb{N} \to \langle \text{symbol} \rangle\,.$$

4(a) Let M be a closed Scheme expression which computes a partial function $f : \mathbb{Z} \to \mathbb{Z}$. Sketch how to transform M into an expression M' such that

1. M' also computes f,
2. M' does not contain any symbols,
3. M' does not contain any occurrence of `int->sym`.

Hint: Represent int-to-sym(n) by pair$(0, n)$, where pair is an integer pairing function. You may assume pair : $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ is total, computable, and one-to-one.

4(b) Conclude that the set of $f : \mathbb{Z} \to \mathbb{Z}$ computable in Scheme extended with a one-to-one `int->sym` is the same as the set of $f : \mathbb{Z} \to \mathbb{Z}$ computable in Scheme *without* `int->sym`.