CBSSS	**Summer 2004**

Computability Problem Set

Out: June 18 *Due: ?*

1. We saw that the language HALT is not decidable, but it is recursively enumerable (or "recognizable"). Show that $\overline{\text{HALT}}$ ("the complement of HALT") defined here:

$$\overline{\text{HALT}} = \{\langle M, w\rangle : M \text{ is a Turing Machine and } M \text{ does not halt on input } w\}$$

   is not recursively enumerable. Hint: prove a general theorem about languages L for which both L and $\overline{L}$ are recursively enumerable.

2. The problem concerns that language TILE, defined as follows. Informally, an instance is a collection of square tiles $t_1, t_2, \ldots, t_k$, together with list of *horizontally compatible* pairs of tiles, and a list of *vertically compatible* pairs of tiles. An $n \times n$ *tiling* is a placement of tiles into an $n \times n$ grid, so that every pair of horizontally adjacent tiles appears in the list of horizontally compatible pairs, and every pair of vertically adjacent tiles appears in the list of vertically compatible pairs; in addition we require that the tile in the upper left corner is tile t_1. The language TILE consists of all those instances for which there is an $n \times n$ tiling for all $n \geq 0$.

   Formally, the language TILE is the set of those tuples

$$\langle k, H = \{(i_1, j_1), (i_2, j_2), \ldots, (i_s, j_s)\}, V = \{(i_1, j_1), (i_2, j_2), \ldots, (i_t, j_t)\}\rangle$$

   for which the following holds. For all $n \geq 0$ there exists a function $f : [n] \times [n] \to [k]$ for which:

   - $f(1, 1) = 1$, and
   - $(f(x, y), f(x, y+1)) \in H$ for all x, and $y \leq n-1$, and
   - $(f(x, y), f(x+1, y)) \in V$ for all y and $x \leq n-1$.

   Here, $[n]$ is shorthand for the set $\{1, 2, 3, \ldots, n\}$. Prove that TILE is undecidable. Hint: it will be helpful to "name" some of your tiles with triplets of symbols.

3. It is obvious that given a reasonable description of an algorithm in pseudo-code, we can produce a Turing Machine that implements that algorithm. What if the pseudo-code included the instruction "print out a description of the Turing Machine running this code"? The self-referential nature of this instruction might lead you to suspect that no Turing Machine could be constructed to implement an algorithm that includes this instruction.

   But, amazingly, this intuition is wrong. It *is* in fact possible to implement such an instruction on a Turing Machine, and this is indeed a very useful and non-obvious fact. The formal statement of this fact is called the Recursion Theorem, stated below:

Theorem 1 (Recursion Theorem) *Let Σ be a finite alphabet, and let T be a Turing Machine that computes the function:*

$$t : \Sigma^* \times \Sigma^* \to \Sigma^*.$$

Then there is a Turing Machine R that computes the function:

$$r : \Sigma^* \to \Sigma^*$$

defined as $r(w) = t(w, \langle R \rangle)$, where $\langle R \rangle$ denotes a description of the Turing Machine R.

(a) Explain why the Recursion Theorem implies the informal statement: it is possible to implement the instruction "print out a description of the Turing Machine running this code" on a Turing Machine.

(b) Use the Recursion Theorem to give a short and simple proof that HALT is undecidable.

(c) Use the Recursion Theorem to give a short and simple proof of Rice's Theorem.