Assignment 2 **Due: Nov 6** **CAS705 Fall 2013**

In this assignment it is important to keep in mind the following: the fact that we can show that an algorithm exists for a given task, does not mean that we know what the algorithm is.

Problem 1. Do exercises 3.1 and 3.9 in the textbook. Then, design a Turing Machine that has the following property $\Sigma = \{1\}$, so there are only two symbols: 1 and $\square$. The machine has 2 states, and takes no input. When started on an empty string, it writes four consecutive 1s, and halts. Can you make it write five consecutive 1s? Can you prove that it is impossible?

Solution: Exercise 3.1 is routine. For exercise 3.9, note that finitely many strings $\{z_1, \ldots, z_n\}$ can always be encoded with states. Furthermore, each one of these strings z_i is a 'yes' or 'no' instance of A_{TM} (though we may not know which one). Now, design M' to be like M, except it also encodes in its states $\{(z_1, a_1), \ldots, (z_n, a_n)\}$, where $a_i = 1$ if z_i is a 'yes' instance, and $a_i = 0$ if z_i is a 'no' instance. On input x, M' checks if $x \in \{z_1, \ldots, z_n\}$, and if $\exists i, x = z_i$, M' answers according to a_i. Otherwise, M' simulates M on x. Clearly M' is a decider for A_{TM} which is not possible.

For the Busy Beaver part of the question, suppose that we have a 2-state 2-symbol machine. Note that the states are $\{q_0, q_1, q_{\text{accept}}, a_{\text{reject}}\}$; the last two states are always there, and q_0 is the initial state. Since the accept/reject states are not really used — their only function is to signal the end of the computation — we shall replace them with q_h, the "halting state." Now consider the following transition function:

	q_0	q_1
$\square$	$(1, q_1, \rightarrow)$	$(1, q_0, \leftarrow)$
1	$(1, q_1, \leftarrow)$	$(1, q_h, \rightarrow)$

Running it on an empty tape (i.e., where each tape cell has a $\square$), so that the tape is infinite in both directions, ends in $\ldots\square 1111\square\ldots$.

I don't have a better proof that five 1s are not possible than an exhaustive search over all possible transition functions: there are $2 \times 3 \times 2$ tuples (2 symbols, 3 states, 2 directions), so 12, and so $12^4 = 20,736$ (in reality less, since we know that at least one of the 4 tuples needs to have q_h in it, since the machine needs to halt). More pruning is possible. Then an examination of each $\ldots$

Problem 2. Suppose that f is a total computable function such that $f(x) \leq f(x+1)$. Show that the range of f, range(f), is a computable set.

Solution: If range(f) is finite, then it is computable (in fact regular). Suppose that range(f) is infinite; then, to check whether y is in range(f), we compute $f(0), f(1), f(2), \ldots$. Since range(f) is infinite, we know that there must exist a k such that $y < f(k)$. Therefore, $y \in \{f(0), f(1), \ldots, f(k-1)\} \iff y \in \text{range}(f)$.

Problem 3. Here are some decision problems involving Turing machines; is it decidable whether a given Turing machine:

1. has al least 1001 states? Decidable: just count that states in Q.
2. takes more than 1001 steps on input ε? Decidable: just run the machine on ε and count the number of steps.
3. takes more than 1001 steps on *some* input? Decidable: given the machine, we simulate it (in sequence) on all inputs of size 1001, for at most 1002 steps. Note that if M takes more than 1001 steps on some input, then it must take more than 1001 steps on some input of length at most 1001 — because in 1001 steps it can read at most the first 1001 symbols of the input.
4. takes more than 1001 steps on *all* inputs? Decidable: similar argument to 3., but we make sure that it runs for more than 1001 steps on all inputs of size at most 1001.
5. ever moves its head more than 1001 tape cells away from the left end on input ε? Decidable: if the machine never moves beyond the first 1001 cells, then it will necessarily start repeating configurations after a while; we can detect this, and conclude that it is in a loop. So either we detect a loop, or the machine will move beyond square 1001.
6. accepts the null string ε? Undecidable: we show that if we could decide whether TMs accept ε, then we could decide A_{TM} as follows: to decide whether M halts on x, construct a new TM M' which has $\langle M, x\rangle$ encoded in its states, and on input ε, it writes $\langle M, x\rangle$ on its tape, and simulates M on x.
7. accepts any string at all? Undecidable: same idea as in 6.; the machine, on any input y, deletes y, and writes $\langle M, x\rangle$, and simulates M on x.
8. accepts every string? Undecidable: same as 7.
9. accepts a finite set? Undecidable
10. accepts a regular set? Undecidable
11. accepts a CFL? Undecidable
12. accepts a decidable set? Undecidable

For the last four languages, let L be finite/regular/CFL/decidable, as needed. The same scheme works in all cases: on input z, the machine saves z, and instead writes $\langle M, x\rangle$ which it had saved in its states, and simulates M on x. If M halts on x, it then simulates the machine for L on y.