CS3 Computability and Intractability (2001-2002)
Exercise Sheet 1

The deadline for this coursework is noon on Thursday 31 January. Please submit your solutions directly to your tutor. Solutions submitted after the deadline, but before noon on Monday 4 February, will have their credit reduced by one third. No credit will be given for solutions submitted after that date. Please note that multiple submissions are not allowed. The marks for questions are not always related to their length or difficulty. Answers that are not completely correct but show relevant reasoning will be awarded partial credit (depending on how much progress is shown). In your answers you should aim for clearness, conciseness and correctness; look at your answers with an objective eye, e.g., imagine that somebody else gave them to you to check.

1. Here we use the notation of Note 1. Consider the *Zero Output* problem:

   Instance: Given natural numbers m, n.

   Question: Does the program P_m when run with input n output 0?

   (a) Describe a program that takes two natural numbers r, s and returns 0 if and only if the program P_r halts on input s. [*5 marks*]

   (b) Is the Zero Output problem computationally solvable, i.e., is there a program Z that that takes arguments m, n and returns True if P_m on input n outputs 0, otherwise it returns False? Explain your answer carefully.

   (You will find it useful to assume you have a computable encoding $\langle r, s\rangle$ of pairs of natural numbers as single natural numbers which can be decoded to recover r, s. You do not have to describe this, there are many easy ways to achieve it.) [*5 marks*]

   Note: The Zero Output problem does *not* ask if P_m halts on n.

2. Design a (one-tape) Turing machine M, with input alphabet $\Sigma = \{0, 1, \$, \#\}$, which has the following behaviour. Let $n > 0$ be a positive integer, and x be a member of $\{0, 1\}^*$. On input $\$x\#^n$, the machine M must eventually halt, leaving $\$x\#^n x$ on its tape. (The action of M on any input which is not of the advertised form is of no interest.)

   Check your design using the Simulator, and submit the result of running M on input `$1011###`.

   Notes: (i) Try to choose 'printing transitions' so as to limit the total output to one or two sides, while providing good evidence that your design is correct. (ii) Five states are enough. (iii) The conclusion, of course, is that a TM is able to copy arbitrary blocks of data across arbitrary distances. [*10 marks*]

3. A two-dimensional Turing machine is one that has a two-dimensional array or *page* of squares, in place of the usual one-dimensional array of squares or *tape*. The page has has a definite top-left corner, but is unbounded below and to the right. The transitions of a two-dimensional machine are similar to those of a conventional one-dimensional machine, except that the tape head can now move one square up, down, left, or right at each step.

   Prove that if a language L is accepted by a two-dimensional Turing machine, then L is accepted by a one-dimensional Turing machine.

   NOTE: A detailed formal proof is *not* expected; however, all the essential steps should be made clear. [*10 marks*]

4. Prove that if there is a Turing machine that accepts the language $L \subseteq \Sigma^*$, then there is a Turing machine that accepts the language

$$L' = \{x \in \Sigma^* \mid x \in L \text{ or } x^R \in L\},$$

   where x^R denotes the reversal of x, i.e., $(x_1 x_2 \cdots x_n)^R = x_n x_{n-1} \cdots x_1$.

   NOTES: Perform a simulation using a TM with two tapes, and appeal to Lemma 4.2. Exercise care—there is a potential pitfall. [*10 marks*]

Kyriakos Kalorkoti, January 2002