UG3 Computability and Intractability (2012-2013)
Exercise Sheet 1: Sample Solutions

Submission date: Friday 19 October.

1. (a) The function f is computable. Given n we can construct P_n and run it on n. If we get a result we just add 1 and return that as the answer.

   Guidelines: 1 mark for knowing that f is computable. 2 marks for the justification. [*3 marks*]

   (b) Suppose that g extends f, is computable and total. Then there is a program P_m that computes g. But now we have a contradiction since by definition the value of $g(m)$ is $P_m(m) + 1$, i.e., $g(m) + 1$. Thus g cannot be computable *and* total.

   Guidelines: Minor variation on bookwork (proof that we cannot have a general system of computation in which all programs halt on all inputs). 2 marks for doing something reasonable with self reference. [*4 marks*]

   (c) Just define

$$g(n) = \begin{cases} f(n), & \text{if } f \text{ is defined on } n, \\ 0, & \text{otherwise.} \end{cases}$$

   This is clearly a total function that extends f. It cannot be computable by the preceding part.

   Guidelines: 2 marks for the definition of g, 2 marks for the explanation that it cannot be computable. The question requires students to use the preceding two parts so no marks if they don't. [*4 marks*]

2. (a) Here is one possible machine:

```
# Given an input string $B@ where B is a binary string the machine
# halts with $B@B on the tape (and accepts the string).

Q = {left, right, zero, one, halt}
I = right
F = halt
S = {0, 1, $, @}
G = {0, 1, A, B, $, @, b}
D = {(right, 0, zero, A, R), (right, 1, one, B, R),
     (right, @, halt, @, L), (right, $, right, $, R),
     (zero, b, left, 0, L), (zero, ?, zero, ?, R),
     (one, b, left, 1, L), (one, ?, one, ?, R),
     (left, A, right, 0, R), (left, B, right, 1, R),
     (left, ?, left, ?, L)}
```

Guidelines: The traces in (b) and (c) should help with checking correctness. Give up to 3 marks for solutions that are correct modulo some minor detail. Give full marks for a solution that is correct, unless it is *far* more complicated than necessary — then give just 4 marks. [*6 marks*]

(b) *Guidelines:* All or none. [*2 marks*]

(c) *Guidelines:* All or none. [*3 marks*]

3. (a) We use a three tape machine. Our first action is to copy the input string on the second tape. On the third tape we list the strings in Σ^* one at a time. After each string is generated we copy the saved input string on the second tape to the first tape and then append the string on the third tape (we do the usual sensible thing about marking the first square of the tapes so that we don't fall off). After this we return to the first (non marked) square on the first tape and simulate the machine that accepts L. This will of course halt. If it accepts we accept otherwise we start the cycle all over.

   Guidelines: Award 6 or 5 marks for a solution that is correct but misses out a detail or two: deduct 1 mark if they fail to mention that the very first squares must be made recognizable, deduct 2 marks if they fail to save the input string in some way. 3 marks for a solution that shows that they are not completely clueless. [*7 marks*]

   (b) The result is still true. We modify the construction above by, e.g., keeping a counter n. At each phase we try the computation of the preceding part for all strings of length at most n and for each attempt we compute for at most n steps (i.e., a version of dovetailing).

   Guidelines: Be generous here: the key point is to realize that the result is still true (1 mark) and that we must try the possibilities with some form of (fake) parallelism. [*4 marks*]

Rahul Santhanam, Thursday 25 October