CS 590 Theory of Computation: Spring 1997
Homework 1

Due 3 February 1997 (Lecture 5)

Problems from the text are denoted by C or BC, depending on whether they are from the Cutland text or from the Bovet and Crescenzi text, followed by the chapter and problem number, separated by a period. For example, C 2.3 3 is problem 3 for chapter 2, section 3 of Cutland.
Question 1 (C 2.3 1a) Without writing any programs, show that for every $m \in \mathbb{N}$ the following functions are computable:

1. $m$ (recall that $m(x) = m$, for all $x$. That is, $m$ is the constant function equal to the number $m$ everywhere.)

2. $mx$ (That is, $f(x) = mx$.)

Answer

1. A program would be easy for $m$, it would zero out R1, then have $m \text{ S}(1)$ instructions. Unfortunately, we’re not supposed to use a program.

   Since every recursive function is computable, I will argue that $m$ is recursive. We have as given that the functions $0(x) = 0$ is computable. We also have that the function $S(x) = x + 1$ is computable. Since the recursive functions are closed under composition, any function $S \circ g \left( S(g(x)) \right)$ is computable when $g$ is. So, if $k$ is recursive, then so is $S \circ k$. But that implies by induction that $m$ is computable for any $m$.

2. We define $f(x) = mx$ for any fixed $m$ by recursion:

   $$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(x - 1) + m & \text{otherwise} \end{cases}$$

   This is a standard definition by recursion, which fully expanded is

   $$f(x) = \begin{cases} g^{(0)} & \text{if } x = 0 \\ h^{(2)}(x - 1, f(x - 1)) & \text{otherwise} \end{cases}$$

   with $g = 0$ (the constant function equal to zero everywhere), and $h(x, y) = y + m$ (where $m$ is fixed). $g$ is recursive by assumption. $h$ is a composition of addition and the constant function $m$ (which we showed recursive above for any $m$). So, $h$ is recursive also. Therefore, $f$ is recursive since it is defined recursively over recursive functions.
Question 2 (C 2.3 2) Suppose that $f(x, y)$ is computable, and that $m \in N$. Show that the function $h(x) = f(x, m)$ is computable.

Answer This problem is deceptively important. In fact, it leads to a central theorem of the theory of computation (the smn theorem).

Suppose that $f = f_P$, that is URM program $P$ computes function $f$. Let $Q$ be a URM program such that $f_Q = m$, that is $Q$ computes the constant function $m$ by always returning $m$ in R1. Now, consider a URM program which calls subroutine $Q$, then transfers the return value from $Q$ (which is $m$) to R2, then calls program $P$. Clearly, such a program computes $h(x) = f(x, m)$. 
**Question 3 (C 2.4 1a,b,f)** Show that the following functions are computable:

a. Any polynomial function $\sum_{i=0}^{n} a_{i}x^{i}$, where each $a_{i}$ is in $N$.

b. $\lceil \sqrt{x} \rceil$. Recall that (page 22, item 1(f)!) $[x]$ is the greatest integer less than or equal to $x$.

f. $\pi(x)$ (this function is usually denote $\pi(x)$ and is called *Euler’s function*—your text denotes it $\phi(x)$) which is the number of positive integers less than $x$ which are relatively prime to $x$. (We say that $x$ and $y$ are *relatively prime* if $HCF(x, y) = 1$.)

**Answer**

a. We prove this by induction.

If $i = 0$, then the polynomial is a constant function $a_{0}$, and is computable by the above arguments.

Suppose $i = k$ and $\sum_{i=0}^{k} a_{i}x^{i}$ is computable. Then $\sum_{i=0}^{k+1} a_{i}x^{i} = a_{k+1}x^{k+1} + \sum_{i=0}^{k} a_{i}x^{i}$.

We know by induction that the right hand term is computable, and addition is computable, so we need to show that $a_{k+1}x^{k+1}$ is computable for any $k$. But this is composition of multiplication of a constant $(a_{k+1})$ with $x^{k+1}$, and we know that multiplication by constants is computable. So we need to show that $x^{k+1}$ is computable. But we know that exponentiation is computable from the text (you could define it by recursion over multiplication).

Therefore $\sum_{i=0}^{k+1} a_{i}x^{i}$ is computable. By induction, this proves that $\sum_{i=0}^{n} a_{i}x^{i}$ is computable for any constants $a_{i}$, which was to be shown.

b. We know that $f(x) = x^{y}$ is computable from the text (proved by recursion over multiplication). An earlier exercise implies that $f(x) = x^{2}$ is computable. Let $P$ be a URM program such that $f_{P}(x) = x^{2}$. Consider the URM program $Q$ for the following pseudocode:

```
Input x
for k=0 to infinity do {
    Call P to compute $y = k^{2}$
    If $y \geq x$ then return $k$ and halt
}
```

Alternatively, one could define $f(x) = \mu y (x - y^{2} = 0)$, and note that the functions inside the $\mu$ operation are themselves recursive.

f. The most compact solution is to define $\pi$ by composition of recursive functions form the text:

$$\pi(x) = \sum_{y=1}^{x-1} \frac{1}{\log(HCF(x, y) - 1)}$$
Question 4 (C 2.5.1) Suppose that $f(x)$ is a total injective computable function. Prove that $f^{-1}$ (the inverse of $f$) is computable.

**Answer** We define $f^{-1}$ using recursiveness-preserving operators, and use the fact that the recursive functions are the computable ones.

$$f^{-1}(x) = \mu y (f(y) = x)$$

Question 5 Thought question: where does your proof in the last question break down if $f$ is not total? Where does it fail if $f$ is not injective?

**Answer** The above proof works fine for non-total $f$. If $f$ is non-total, then the range of $f^{-1}$ is simply not all of $N$.

If $f$ is not injective, then there are at least two $x, y$ with $x \neq y$ such that there is a $z$ with $f(x) = f(y) = z$. In this case, $f^{-1}$ is not a function at all, since $f^{-1}(z)$ has two possible values. Note, though, that the function we defined above would still be a good one. That is $g(x) = \mu y (f(y) = x = 0)$ would be well defined and in fact $g(x)$ would be the least value $y$ such that $f(y) = x$.

Question 6 Let the $A(n) = \psi(n, n)$, for Ackerman’s function $\psi$ on page 46 of Cutland. Make a table for $A(n)$ for as many values as you can stand.

**Answer** I’ve suffered enough. It’s your turn.