CSE 321: Discrete Structures
Assignment #6
May 15, 2002
due: Wednesday, May 22

1. It is a somewhat amazing fact that the greatest common divisor can be written as a linear combination, that is, $\gcd(a, b) = sa + tb$, for some integers s and t. It is sometimes important to be able to compute not only the greatest common divisor, but the coefficients s and t as well. (Part (b) of this problem gives an example application.) The following extension of Euclid's algorithm computes the gcd g plus those coefficients. Try it out on some examples.

   (The programming notation $(x, y) \leftarrow (e, f)$ means simultaneous assignments of the old value of e to x and the old value of f to y. For instance, the body of the ordinary Euclidean algorithm's loop could have been written $(x, y) \leftarrow (y, x \bmod y)$. Note that this is exactly the effect of the statement $(a_0, a_1) \leftarrow (a_1, a_0 - q * a_1)$ below, so that the output g is still $\gcd(a, b)$.)

   procedure Extended_Euclid $(a, b$: integer$)$ **returns** g, s, t: integer
   begin
   $(a_0, a_1) \leftarrow (a, b)$;
   $(s_0, s_1) \leftarrow (1, 0)$;
   $(t_0, t_1) \leftarrow (0, 1)$;
   while $a_1 \neq 0$ **do**
   begin
   $q \leftarrow \lfloor a_0/a_1 \rfloor$;
   $(a_0, a_1) \leftarrow (a_1, a_0 - q * a_1)$;
   $(s_0, s_1) \leftarrow (s_1, s_0 - q * s_1)$;
   $(t_0, t_1) \leftarrow (t_1, t_0 - q * t_1)$;
   end ;
   $g \leftarrow a_0$;
   $s \leftarrow s_0$;
   $t \leftarrow t_0$;
   end .

   (a) Prove that the inputs and outputs satisfy $g = sa + tb$. (Hint: Use induction to prove that $a_0 = s_0 a + t_0 b$ and $a_1 = s_1 a + t_1 b$ at the beginning of each iteration.)

   (b) The *inverse* of a mod m, if it exists, is an integer s such that $as \equiv 1 \pmod m$. As an example of the usefulness of this algorithm, show that whenever $\gcd(a, m) = 1$, the outputs of Extended_Euclid(a, m) produce an inverse of a mod m. (It turns out that an inverse of a mod m only exists when $\gcd(a, m) = 1$. It's not a hard proof, if you feel like trying it.)

2. A *binary tree* is either empty, or consists of a root node and a "left subtree" and "right subtree", which are themselves binary trees with no nodes in common. (See Figure 8 in Section 8.1 for an example.) Any node in a binary tree both of whose subtrees are empty is called a *leaf.* For example, the tree in Figure 8(a) of Section 8.1 has 6 leaves: f, g, e, j, k, m. The *height* of a binary tree is the distance from the root to the farthest leaf. The tree in Figure 8(a) of Section 8.1 has height 4, m being the farthest leaf from the root. (Note that the distance from the root to m is considered to be 4 rather than 5: it's the number of edges on the path, rather than the number of nodes.) By induction, prove that for any positive integer n, any binary tree with n leaves has height at least $\log_2 n$. Be careful of the possibility that a node has one empty subtree and one nonempty subtree. (Hint: it will be simplest if your induction mirrors the recursive definition of binary tree given above.)

3. Section 3.3, exercise 28. I don't know what is meant by a "recursive proof"; instead, use induction on the length $|w_2|$. I want you to use the recursive definition of reversal given in exercise 27, rather than the more imprecise definition given before exercise 26.