Math 55: Discrete Mathematics, Fall 2008
Homework 2 Solutions

Problems marked * will be corrected fully, out of 10 points. The others will be checked quickly, for 2 points each.

2.3: 16. The following are not the only possible solutions but they are some of the simplest ones.
(a) \( f(n) = n + 1 \).
(b) \( f(0) = 0, f(n) = n - 1 \) if \( n > 0 \).
(c) \( f(0) = 1, f(1) = 0, f(n) = n \) if \( n > 1 \).
(d) \( f(0) = 1, f(n) = n \) if \( n > 0 \).

40. (a) \( f^{-1}(S \cup T) = \{ a \in A : f(a) \in S \cup T \} = \{ a \in A : f(a) \in S \text{ or } f(a) \in T \} = \{ a \in A : f(a) \in S \} \cup \{ a \in A : f(a) \in T \} = f^{-1}(S) \cup f^{-1}(T) \).
(b) Just change \( \cup \) to \( \cap \) and “or” to “and” in the above solution to part (a).

A more laborious but also correct way to do this is to verify in each part that the set on the left-hand side is a subset of the one on the right, and conversely.

*[5 pts each part] (C) (i) It is true that if \( f \circ g \) is one-to-one, then \( g \) is one-to-one. To prove it, suppose \( g(x) = g(y) \). Then \( f \circ g(x) = f \circ g(y) \), which implies \( x = y \) since \( f \circ g \) is one-to-one. We have shown that \( g(x) = g(y) \) implies \( x = y \), so \( g \) is one-to-one.

(ii) It is false that if \( f \circ g \) is one-to-one, then \( f \) is one-to-one. For a counterexample, define \( g : \{1\} \to \{1, 2\} \) by \( g(1) = 1 \), and define \( f : \{1, 2\} \to \{1\} \) by \( f(1) = f(2) = 1 \). Then \( f \circ g \) is the identity function on \( \{1\} \), which is one-to-one, but \( f \) is not one-to-one.

2.4: 32(b) \( S = \{-1, -3, -5, \ldots\} \) is countable, with an explicit bijection \( f : \mathbb{N} \to S \) given by \( f(n) = -2(n + 1) \).

(c) Uncountable. It has the same cardinality as the set of all real numbers.

*40. Suppose given bijections \( f : \mathbb{N} \to S \) and \( g : \mathbb{N} \to T \). We can define an onto function \( h : \mathbb{N} \to S \cup T \) by \( h(n) = f(n/2) \) if \( n \) is even, \( h(n) = f((n-1)/2) \) if \( n \) is odd. In other words, if the enumeration of \( S \) given by \( f \) is \( \{s_0, s_1, s_2, \ldots\} \) and that of \( T \) given by \( g \) is \( \{t_0, t_1, t_2, \ldots\} \), then we enumerate \( S \cup T \) in the order \( \{s_0, t_0, s_1, t_1, s_2, t_2, \ldots\} \).

If \( S \) and \( T \) are not disjoint, the function \( h \) is not one-to-one. To fix this, define \( g(n) \) to be the first element on the list \( \{s_0, t_0, s_1, t_1, s_2, t_2, \ldots\} \) that is not in the set \( \{g(1), g(2), \ldots, g(n-1)\} \). Note that there is always such an element because \( S \cup T \) is infinite. Then \( g \) is one-to-one by construction. Given any \( x \in S \cup T \), let \( n \) be the number of distinct elements on the list up to and including the first occurrence of \( x \). Then \( g(n) = x \), which shows that \( g \) is onto.

3.1: 24. Here is one possibility. Assume that the function is \( f : A \to B \) and that the input consists of a list \( a_1, \ldots, a_m \) of the elements of \( A \), a list \( b_1, \ldots, b_n \) of the elements of \( B \), and a list \( f(a_1), \ldots, f(a_n) \) of the function values.

\[
\text{input } a_1, \ldots, a_m; b_1, \ldots, b_n; f(a_1), \ldots, f(a_n) \\
\text{for } i = 1 \text{ to } n \text{ do:} \\
\quad k = 0 \text{ [this } k \text{ will count the number of elements mapped to } b_i] \\
\quad \text{for } j = 1 \text{ to } m \text{ do:}
\]

Homework 2 Solutions
if \( f(a_j) = b_i \) then \( k = k + 1 \)
if \( k > 1 \) return “Not one-to-one”
next \( j \)
next \( i \)
return “One-to-one” [the algorithm only reaches this point if \( f \) is one-to-one]

60. Suppose this were solvable by an algorithm \( S \). Then we could solve the halting problem for an algorithm \( A \) on input \( I \) as follows: From \( A \) construct algorithm \( B_A \) that runs \( A \) (discarding any output), and if \( A \) terminates, outputs 1. Then \( S(B_A, I) \) solves the halting problem for \( A \) on input \( I \).

*(A) The following is also unsolvable: given an algorithm \( A \) which is known in advance to halt on every input, to decide whether there exists an input \( I \) such that \( A \) eventually outputs the symbol “1” when run with input \( I \).

Suppose this were solvable by an algorithm \( T \). Then we could solve the halting problem for an algorithm \( B \) on input \( J \) as follows. From \( B \) and \( J \) construct algorithm \( A_{B,J} \) that takes as input a positive integer \( n \), runs \( B \) on \( J \) for \( n \) steps, then halts and outputs 1 if (and only if) \( B \) has halted by then. Note that \( A_{B,J} \) is known to halt on every input. Now run \( T \) on \( A_{B,J} \). Then \( T \) concludes that there exists an input \( n \) on which \( A_{B,J} \) halts, if and only if \( B \) halts on \( J \).

3.4: 4. Suppose \( a \mid b \) and \( b \mid c \). Then there are integers \( k \) and \( l \) such that \( b = ka \) and \( c = lb \). It follows that \( c = (kl)a \), so \( a \mid c \).

8. Counterexample: Take \( a = 6, b = 2, c = 3 \). Then \( a \mid bc \) (since of course 6 divides itself), but \( a \) does not divide \( b \) or \( c \).

*22. The congruence \( a \equiv b \) (mod \( m \)) means that \( m \mid b - a \). Say \( b - a = km \). Then \( bc - ac = kmc \), which shows that \( mc \mid bc - ac \), and thus \( ac \equiv bc \) (mod \( mc \)). [The hypothesis \( m \geq 2 \) in the problem is not necessary; the result is also valid for \( m = 1 \). Of course one always assumes \( m > 0 \) when working with congruences mod \( m \).]