MC 215 MATHEMATICAL REASONING AND DISCRETE STRUCTURES, FALL 2007
Solutions to Hand-in Homework #4, 11/10/08 – 47 points total

1. p. 158 #34 – for each of the five properties asked about, prove that your answer is correct (e.g., if you say it's reflexive, prove it; if you say it's not, give a counterexample). **(15 pts)**

Solution to #1: X = set of 4-bit strings, and x R y means that some 2-bit substring of x equals a 2-bit substring of y, i.e., they have a 2-bit substring in common.

Reflexive? **YES**. For any $x \in X$, x R x, if $x = x_1x_2x_3x_4$, then the 2-bit substring x_1x_2 trivially equals itself.
Symmetric? **YES.** For any x, $y \in X$, if x R y, then some 2-bit string is a substring of both x and y, hence also of y and x, which implies y R x.
Antisymmetric? **NO**. Let x = 0000 and y = 0011. Then x R y and y R x, since they both contain the 2-bit substring 00, but $x \neq y$.
Transitive? **NO.** Let x = 0000, y = 0011, and z = 1111. Then x Ry, because they both have 00 as a substring, and y R z, since they both have 11 as a substring, but it's not true that x R z, since they have no 2-bit substring in common.
Partial order? **NO,** because R is neither antisymmetric nor transitive.

2. p. 165 #32 **(14 pts: a-9, b-4, c-1)**

Solution to #2: X = {1, 2, 3, 4, 5} and R is the relation on X x X defined by (a, b) R (c, d) if and only if ad = bc. Note that this is equivalent to saying a/b = c/d, which I'll use instead.

a) To show that R is an equivalence relation, we must show that it is reflexive, symmetric, and transitive.
R is reflexive: a/b = a/b, so (a, b) R (a, b).
R is symmetric: If (a, b) R (c, d) then a/b = c/d. This implies c/d = a/b, thus (c, d) R (a, b).
R is transitive: If (a, b) R (c, d) and (c, d) R (e, f), then a/b = c/d and c/d = e/f. This implies a/b = e/f, thus (a, b) R (e, f).
From the definition of equivalence relation, we conclude that R is an equivalence relation.

b) There are 19 equivalence classes – only 3 of them have more than one element. They are listed below, with a representative chosen for each shown in the square brackets.

[(1, 1)] = {(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)}
[(1, 2)] = {(1, 2), (2, 4)}
[(1, 3)] = {(1, 3)}
[(1, 4)] = {(1, 4)}
[(1, 5)] = {(1, 5)}
[(2, 1)] = {(2, 1), (4, 2)}
[(2, 3)] = {(2, 3)}
[(2, 5)] = {(2, 5)}
[(3, 1)] = {(3, 1)}
[(3, 2)] = {(3, 2)}
[(3, 4)] = {(3, 4)}
[(3, 5)] = {(3, 5)}
[(4, 1)] = {(4, 1)}
[(4, 3)] = {(4, 3)}
[(4, 5)] = {(4, 5)}
[(5, 1)] = {(5, 1)}
[(5, 2)] = {(5, 2)}
[(5, 3)] = {(5, 3)}
[(5, 4)] = {(5, 4)}

c) In familiar terms, (a, b) R (c, d) if and only if the fractions a/b and c/d are equal.

3. p. 185 #6. In addition to writing the pseudocode for this problem, you should *trace* it on the following list of numbers: 32, 2, 23, 40, 17. Do this by numbering each line of your algorithm, and then showing, in order, which line is executed and which objects (the list, the counter, or other variables you use) are changed. **(12 pts: code-6, trace-6)**

Solution to #3: "<u>big1</u>" records the current biggest value, and <u>big2</u> records the current 2nd biggest value.

1. *max2(s,* n)	Before algorithm starts: s = {32, 2, 23, 40, 17}
2. big1 = s_1	Line 2) big1 = 32
3. if (s_2 > big1) then	Line 3) (2 > 32) false
4. big1 = s_2	Line 7) big2 = 2
5. big2 = s_1	Line 9) i = 3
6. else	Line 10) (23 > 32) is false
7. big2 = s_2	Line 13) (23 > 2) is true
8. end if	Line 14) big2 = 23
9. for i = 3 to n	Line 9) i = 4
10. if (s_i > big1) then	Line 10) (40 > 32) is true
11. big2 = big1	Line 11) big2 = 32
12. big 1 = s_i	Line 12) big1 = 40
13. else if (s_i > big2) then	Line 9) i = 5
14. big2 = s_i	Line 10) (17 > 40) is false
15. end if	Line 13) (17 > 32) is false
16. end for	Line 17) return 40, 32
17. return big1, big2	
18. end max2	

4) Write the pseudocode for Bubble Sort on a list *s = {s_1, … s_n}* with *n* elements. **(6 pts)**

Solution to #4: "<u>last</u>" is the index of the last element in the unsorted section of the list, and "<u>i</u>" is the index of the element that is being compared to the next element (with index i + 1).

BubbleSort(s, n)
for last = n down to 2
for i = 1 to last -1
if (s_i > s_{i+1}) then
swap(s_i, s_{i+1})
end if
end for
end for
end BubbleSort