CSE3401 Summer 2009 Assignment #2

Due by July 13dth 2009 11:59PM

Question #1 (50 points)

Write a recursive function COMPRESS and DECOMPRESS that takes a list as a parameter and replaces any consecutive occurrence of elements with the element and its count. For example:

(compress '(a a a b b x 2 2))
➔ (a 3 b 2 x 1 2 2)
(decompress '(a 3 b 2 x 1 2 2))
➔ (a a a b b x 2 2)

Modify the COMPRESS and DECOMPRESS functions and define new functions COMPRESS2 and DECOMPRESS2 to compress the nodes of the list if they are lists as well. For example:

(compress2 '(a a a (b b) (b b) x 2 2))
➔ (a 3 (b 2) 2 x 1 2 2)
(decompress2 '(a 3 (b 2) 2 x 1 2 2))
➔ (a a a (b b) (b b) x 2 2)

Note: Use helper functions and comments to make your code clear to understand. Perform thorough testing of your functions and include in your answer.

Correct solutions can get up to 40 points, and well-written and well-tested solutions will get up to 10 additional points.

Question #2 (20 points)

Write a recursive LISP function to test if a given s-expression is a proper LISP list and not just a dotted-pair expression that has no list representation. For example:

(listpp '(a b c))
➔ (a b c)
(listpp '(a b . c))
➔ nil
(listpp '(a (b . c) d))
➔ nil

Note: Use comments to make your code clear to understand. Perform thorough testing of your functions and include in your answer. You can use helper functions. Be careful how you handle NILs since they are lists but not conses.

Note: LISP built-in LISTP function does not really do what is described in this question. Show the differences.

Correct solutions can get up to 15 points, and well-written and well-tested solutions will get up to 5 additional points.

Question #3 (20 points)

Common LISP guarantees that the arguments to a function will be evaluated in sequence but does not specify if it is left-to-right or right-to-left. You need to develop a method and the determine the order of argument evaluation in your version of Common LISP. Remember to indicate in your answer what version and platform is your Common LISP.

Question 4 (25 points)

Modify the INNER-PRODUCT functional as INNER-PRODUCT2 to take 2 additional functions INNER-FUNC and OUTER-FUNC as arguments. These functions abstractions of the inner product internal operations and are to be applied in the inner product computation in place of * and +.

Now how would you compute a vector multiplication where elements of the vector are also vectors? For example:

[[a1 a2 a3][b1 b2 b3]] x [[c1 c2 c3][d1 d2 d3]] =
[a1 a2 a3] x [c1 c2 c3] + [b1 b2 b3] x [[d1 d2 d3] =
a1*c1+a2*c2+a3*c3 + b1*d1+b2*d2+b3*d3

Note: Use helper functions and comments to make your code clear to understand. Perform thorough testing of your functions and include in your answer.

Correct solutions can get up to 20 points, and well-written and well-tested solutions will get up to 5 additional points.