CSE 3401- Summer 2010
Functional Programming- Review Questions

Assume we have entered the following expressions in the LISP interpreter:
> (setq x 5)
5
> (setq lst '(1 2 3 4))
(1 2 3 4)
> (setq fname #'(lambda (x) (* 10 x)))
#<FUNCTION :LAMBDA (X) (* 10 X)>
> (setq gname #'(lambda (x) (cons x 'x)))
#<FUNCTION :LAMBDA (X) (CONS X 'X)>

How would LISP respond to the following?

> (car lst)
1
> (cdr lst)
(2 3 4)
> (cadr lst)
2
> (fname lst)
(10 20 30 40)
> (fname (car lst))
10
> (apply fname (car lst))
10
> (apply fname lst)
(10 20 30 40)
> (apply fname (list (car lst)))
(10)
> (mapcar fname lst)
(10 20 30 40)
> (mapc fname lst)
(10 20 30 40)
> (1 . nil)
(1 . nil)
> '(1 . nil)
'(1 . nil)
Use cond to write a function \( f_1 \) as follows:
\[
f_1(x) = \begin{cases} 
-1 & x < 0 \\
1 & 0 \leq x < 10 \\
2 & 10 \leq x < 30 \\
3 & x \geq 30
\end{cases}
\]

Use cond to write a function \( f_2 \) with two arguments \( x \) and \( \text{lst} \) that does the following:
- If \( x \) is a negative number, it opens the file "data.txt", reads from it once and returns the read number (we'll assume it will be a number) as string containing the number as a float with 2 digits after the decimal point.
- If \( x \) is zero, it returns true
- If \( x \) is a positive number, it returns the first two elements of \( \text{lst} \) (we assume \( \text{lst} \) has at least two elements)
- If \( x \) is anything else, it returns nil

If \( f_3 \) is defined as follows, how would LISP respond to the following?
\[
\text{(defun } f_3 \text{ (lst n p)} \\
\text{ (do ((tlst lst (cdr tlst))} \\
\text{ (rslt ')(0 . nil) (cons (car tlst) rslt))} \\
\text{ (i (1- n) (1- i)))}
\]
((zerop i) (cond ((zerop p) rslt) (t n)))
(if (null tlst) (return "Error")))

> (f3 '(1 2 3) 3 0)
> (f3 '(1 2 3) 3 1)
> (f3 '(1 2 3) 5 1)
> (f3 '(1 2 3) 5 0)
> (f3 '(1 2 3) 4 0)

What if do was replaced with do*?

================================================================

Alonzo Church has defined the natural numbers in lambda calculus (known as the Church numerals) as follows:

0 := λfx.x
1 := λfx.f x
2 := λfx.f (f x)
3 := λfx.f (f (f x))

Show that if PLUS is defined as
PLUS := λmnfx.m f (n f x)
then adding (or PLUS) 2 and 1 is equivalent to 3.

(Try AND or NOT in logical predicates, or multiplication in arithmetic, see Wikipedia)

================================================================

[ref: CSE3401 Summer 2009 Assignment #2]
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)

Write a function that
- Creates a sequence of bits (0 or 1) of length len.
- Convert a sequence of bits to its decimal equivalent:
- Write a function that inverts a random bit in a sequence with a given probability.