LISP Coding Exercises

1. Code the recursive function, **SETDIFF**, which is passed two sets which are all top-level items. It returns the difference of set1 and set2 (i.e., it returns items in set1 which are not in set2).

Examples:

(setdiff '(x y z) '(w x z)) is (Y)

(setdiff '(w x y z) '(x y)) is (W Z)

(setdiff '(x y z) '(x y z)) is ()

(defun setdiff (set1 set2)

(cond

((NULL set1) ())

((memset (car set1) set2)

(setdiff (cdr set1) set2))

(T (cons (car set1) (setdiff (cdr set1) set2)))

Recursive means no DO loop

2. Code the recursive function (**replaceIn** *list possibleList* *repValue*) which constructs a new list. It examines *list* for occurrences of any of the atoms from the *possibleList*. Those are replaced with *repValue.* This only examines the top-level items in *list*.

Example:

> (**replaceIn** '(P A T T E R) '(T R) 'S)

(P A S S E S)

(defun replaceIn (L possibleList repValue)

(cond ( (NULL L) () )

(( memset (car L) possibleList) (cons repValue (replaceIn (cdr L) possibleList repValue)))possibleList repValue)) )

( T (cons (car L) (replaceIn (cdr L) possibleList repValue)) )

)

(defun replaceIn( L possibleList repValue)

(cond

( (NULL L) NIL )

3. Code the LISP macro, (**IsIn** *value* arg1 arg2 …), which is passed a *value* and a variable number of arguments. It returns T if *value* matches one of the **evaluated** arguments. If none of the evaluated arguments matches *value*, **IsIn** returns NIL.

To refresh your mind about macros, here is the code from the notes on the switch macro:

(defmacro switch (value &rest pairs)

(let ((g (gensym)) )

`( do ((,g ' ,pairs (cdr ,g)) )

((null ,g) NIL)

(if (or

(eql 'DEFAULT (caar ,g))

(eql ,value (eval (caar ,g)))

)

(return (eval (cadar ,g)))

) ) ) )

A. Examples based on the problem description:

> (**IsIn** 'X 'Y 'Z)

NIL

> (**IsIn** 'X (cons 'X NIL) (car (list 'X)) )

T

Let’s try doing a function first

(defun IsIn(value &rest args)

(do ( (cv args (cdr cv)) )

( (null cv) NIL)

(if (eql value (car args))

(return T) )

) )

(defMacro IsIn (value &rest args)

(let ( (gcv (gensym)) )

`do ( (,gcv ‘,args (cdr ,gcv)) )

( (null ,gcv) NIL )

(if (eql ‘,value (eval (car ,gcv)))

(return T) )

) )

B. Suppose we wanted the first argument to be passed in as unevaluated:

> (**IsInv2** X 'Y 'Z)

NIL

> (**IsInv2** X (cons 'X NIL) (car (list 'X)) )

T

(defMacro IsInv2 (