**Advanced Programming Practices**

**Assignment 2**

**40233170: ARYAN SAXENA**

**40197012: ANANYA VARSHA**

--------------------------------------------------------------------------------------------------------------------------------------------------

**1. Implement function swap which takes as an argument a list and returns a new list which represents the argument list where each two consecutive elements are swapped.**

(defparameter temp '())

(defun swap(lst)

(cond ((null lst) nil)

((equal (list-length lst) 1) (setq temp (append temp (list (car lst)))))

(t (setq temp (append temp (list (cadr lst))))

(setq temp (append temp (list (car lst))))

(setq lst (delete (car lst) lst))

(setq lst (delete (car lst) lst))

(swap lst))))

(swap '(1 "two" -3 "four"))

(write temp)

**OUTPUT:**

Text

Description automatically generated

Shape

Description automatically generated with medium confidence

--------------------------------------------------------------------------------------------------------------------------------------------------

**2. Provide a) a recursive definition and b) the implementation of function compress which takes a list as its argument and returns a new list where all consecutive duplicates of its argument are replaced with a single copy of the element. The order of the elements should not be changed.**

(defun compress (L)

(cond ((null L) L)

((equalp (first L)

(second L))

(compress (cdr L)))

(t (cons (car L) (compress (cdr L))))))

**OUTPUT:**

Text

Description automatically generated

--------------------------------------------------------------------------------------------------------------------------------------------------

**3. Provide the implementation of function pairs that returns a list of pairs of corresponding elements from two lists of equal length.**

(defvar length1)

(defvar length2)

(setq \*print-case\* :downcase)

(defun pairs(list1 list2)

(setq length1 (list-length list1))

(setq length2 (list-length list2))

(if (= length1 length2)

(loop for x in list1 for y in list2 collect (list x y))))

**OUTPUT:**

Text

Description automatically generated

--------------------------------------------------------------------------------------------------------------------------------------------------

**4. Provide the implementation of function dist that accepts an atom n and a non-empty list lst, and returns a list composed of lists of two elements, the first being n and the second being each successive element of lst.**

(defvar length1)

(setq \*print-case\* :downcase)

(defun dist(n lst)

(setq length1 (list-length lst))

(loop for x in lst collect (list n x)))

**OUTPUT:**

Text

Description automatically generated

--------------------------------------------------------------------------------------------------------------------------------------------------

**5. Implement function diff that takes two non-empty lists of equal length as arguments and produces a list whose elements correspond to the cubed differences between the corresponding elements of the two arguments. We may assume that non-empty list arguments contain only numeral elements. Example executions are as follows.**

(DEFUN DIFF (A B)

(SETQ C ( MAPCAR #'- A B))

(SETQ C (MAPCAR #'\* C C C))

(SETQ X ( LIST-LENGTH A))

(SETQ Y ( LIST-LENGTH B))

(IF (= X Y)

C

NIL))

**OUTPUT:**

Text

Description automatically generated

-------------------------------------------------------------------------------------------------------------------------------------------------

**6. Implement function (list2set lst) that takes a list as an argument and returns a set representation of the list. Example executions are as follows:**

(DEFUN LIST2SET (A)

(SETQ B (LIST-LENGTH A))

(IF(> B 1)

(REMOVE-DUPLICATES A)

))

**OUTPUT:**

Text

Description automatically generated with medium confidence

--------------------------------------------------------------------------------------------------------------------------------------------------

**7. Implement function (filter list numeral) that takes two arguments, a) a non-empty list of integers, and b) a positive integer, and produces a list whose elements are those elements of the first argument that are larger than the second argument. We may assume that a non-empty list argument contains only numeral elements.**

(defun filter(lst1 lst2)

(cond ((null lst1) ())

((<= (car lst1) lst2 ) (filter (cdr lst1) lst2))

(t (cons (car lst1) (filter (cdr lst1) lst2)))

))

**OUTPUT:**

