```
"""#-----------------------------------------------------------------------------#"""
"""#---------------------      METU Cognitive Sciences         ---------------------#"""
"""#---------------------          Turgay Yıldız               ---------------------#"""
"""#---------------------      yildiz.turgay@metu.edu.tr       ---------------------#"""
"""#-----------------------------------------------------------------------------#"""



"""#-----------------------------------------------------------------------------#"""
"""#---------------------          Exercise 6.1                ---------------------#"""
"""#-----------------------------------------------------------------------------#"""

    (defun  vals    (list1    arg)

                (if     (endp list1)
                        nil
                        (cons   (funcall    (car list1)   arg)   (vals  (cdr list1)  arg) )
                )

    )

    ; (vals '(evenp log zerop)  8)      will return     (T    2.0794415   NIL)


"""#-----------------------------------------------------------------------------#"""
"""#---------------------          Exercise 6.2                ---------------------#"""
"""#-----------------------------------------------------------------------------#"""

    (defun  pairvals_   (list1    args   func_list)      ;  (pairvals '(f g h)  8)    should give   ( (F. 64)  (G. 512)  (H. 2.079234))

            (if     (endp list1)
                    nil
                    (append     (list (cons (car list1)  (funcall (car  func_list) args)))      (pairvals_ (cdr list1) args (cdr func_list)))
            )



    )

    (defun func1 (x) (* x x))
    (defun func2 (x) (* x x x))

    (defun  pairvals    (list1   args)

            (pairvals_  list1  args  '(func1 func2 log) )
        )

"""#-----------------------------------------------------------------------------#"""
"""#---------------------          Exercise 6.3                ---------------------#"""
"""#-----------------------------------------------------------------------------#"""



    (defun  maxpair   (y  &optional (storage '(a . 0)))

        (if     (endp y)
                storage

                (if (< (cdr (car y)) (cdr storage))

                    (maxpair   (cdr y)   storage)
                    (maxpair   (cdr y)   (car y) )
                )
        )

    )

"""#-----------------------------------------------------------------------------#"""
"""#---------------------          Exercise 6.4                ---------------------#"""
"""#-----------------------------------------------------------------------------#"""

(defun      func    (list_pred  object   storage_success)       ; (func '(consp listp numberp) '(a) nil)    ->      (LISTP CONSP)

    (cond   ( (endp list_pred)                                      storage_success)
            ( (funcall (car list_pred)  object)                     (func (cdr list_pred) object (cons (car list_pred) storage_success)))
            ( t                                                     (func (cdr list_pred) object storage_success))

    )
)



"""#-----------------------------------------------------------------------------#"""
"""#---------------------          Exercise 6.5                ---------------------#"""
"""#-----------------------------------------------------------------------------#"""

(defun odd_p (x)
    (cond   ( (and (integerp x) (oddp x))                   t)
            ( t                                             nil)
    )
)

(defun even_p (x)
    (cond   ( (and (integerp x) (evenp x))                  t)
            ( t                                             nil)
    )
)

(defun      func_   (list_pred  object)             ; (func   '(odd_p even_p)    12  )

            (cond   ( (endp list_pred)                                      nil)
                    ( (funcall (car list_pred)  object)                     t )
                    ( t                                                     (func_  (cdr list_pred)  object) )
            )
)


(defun      func    (list_pred   list_object  storage_success)   ; (func   '(odd_p even_p)    '(12 23  12.4  15.3 16 12.3)  nil)
```

```
            (cond   ( (endp list_object)                                    storage_success)
                    ( (func_ list_pred (car list_object))                   (func  list_pred   (cdr list_object)  (cons (car list_object) storage_success)))
                    ( t                                                     (func  list_pred   (cdr list_object)   storage_success))
            )
    )


; Any object that is not suitable for any function will give error. E.g., (oddp  12.3)

    """#-----------------------------------------------------------------------------------#"""
    """#---------------------          Exercise 6.6               ----------------------#"""
    """#-----------------------------------------------------------------------------------#"""


    (defun square (x) (* x x))
    (defun cube   (x) (* x x x))
    (defun func   (x) (* 2 (* x x)))

    (defun      find_max  (pred_list  x )       ; (square   cube   sqrt   ... )    (3)   (27)

                (if     (endp pred_list)
                        0
                        (max  (funcall (car pred_list)  x)    (find_max   (cdr pred_list)  x ))
                )
    )


    """#-----------------------------------------------------------------------------------#"""
    """#---------------------          Exercise 6.7               ----------------------#"""
    """#-----------------------------------------------------------------------------------#"""

    (defun   func2   (lst    pred  index  storage)        ; (func2  '(0 1 1 0 0 1 0)   #' zerop 0 nil)   ->   (0 3 4 6)

            (cond   ( (endp lst)                            (reverse storage))
                    ( (funcall pred (car lst))              (func2 (cdr lst) pred (+ index 1) (cons index storage)))
                    ( t                                     (func2 (cdr lst) pred (+ index 1)             storage))
            )
    )
```