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



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

Define a procedure VALS that takes a list of one argument procedures and an argu-
ment, and returns the values obtained by applying the procedures to the argument
in the given order.

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


(defun  vals    (list1    arg)

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

)

"""#---------------------          Second Way              ---------------------#"""

(defun  vals    (list1    arg   storage)

            (if     (endp list1)
                    (reverse storage)
                    (vals  (cdr list1)  arg   (cons (funcall (car list1) arg) storage) )
            )

)


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

Define a procedure PAIRVALS that takes a list of one argument procedures and an
argument, and returns the list of dotted pairs where each procedure is paired with
the value obtained by applying it to the argument.

(pairvals '(f g h)  8)    should give   ( (F. 64)  (G. 512)  (H. 2.079234))


(defun  pairvals_   (list1    args   func_list)

        (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                ---------------------#"""
"""#----------------------------------------------------------------------------------#"""

Define a procedure MAXPAIR that takes a list of dotted pairs and returns the maxi-
mum pair where the comparison is done on the basis of the second components of
pairs.

* (maxpair '((A . 2) (B . 8) (C . 4)))

(B . 8)

Note that :

* (cdr '(a . 0)  )
0

* (cdr '(a 0)  )
(0)


(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) )
            )
    )
)

"""#---------------------          Second Way              ---------------------#"""


(defun maxpair2 (lst)

    (reduce #' (lambda (x y)
```

```
        (if     (> (cdr x) (cdr y))
                x
                y
        ))
        lst
    )
)

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

Define a procedure that takes a list of predicate symbols (e.g. CONSP, NUMBERP etc.)
and an object, and returns the list of predicates that the object satisfies. Here is a
sample interaction:

(foo '(consp listp numberp) '(a)) ==> (CONSP LISTP)


(defun  func (list_pred  object   storage_success)

    (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))
    )
)

"""#---------------------          Second Way             ----------------------#"""

(defun func4 (pred obj)

    (mapcar #' (lambda (x)

        (if     (funcall x obj)
                x
                nil
        ))
    pred)
)


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

Define a procedure that takes a list of predicate symbols (e.g. CONSP, NUMBERP etc.)
and a list of objects, collects and returns all the objects that answer yes to at least
one predicate in the predicate list.


(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)

"""#---------------------          Second Way             ----------------------#"""

(defun func5 (list_pred   list_object result)

    (mapcar #' (lambda (obj)

        (dolist (i list_pred result)

                (if (funcall i obj)
                    (setf result obj)
                    nil
                )
        ))
        list_object)
)

Note: (funcall with #') will not work . Because it will be detected as (funcall #'')

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

Define a procedure that takes a list of one argument numerical procedures (define
your own and/or use the built-ins you know) and a number, and returns the name
of the procedure that yields the maximum value when applied to the number argu-
ment.
```

```
 (square   cube   sqrt   ... )    (3)   (27)


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

(defun      find_max  (pred_list  x )

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


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

Take a list and a procedure as input and return the list of indices of all the elements
that the procedure returns a non-nil value.

(func2  '(0 1 1 0 0 1 0)   #' zerop  )   ->   (0 3 4 6)


(defun   func2   (lst    pred  index  storage)

    (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))
    )
)


"""#-----------------------------------------------------------------------------------#"""
"""#---------------------                END                ----------------------#"""
"""#-----------------------------------------------------------------------------------#"""
```