MiniAsgn1

Kushagra Indurkhya

03 March 2021

Instruction:

- You have to submit 10 simple programs in Lisp/Scheme
- Sections are created below, you have to write the Lisp code, comments in the code will be rewarded, an analysis your code, and explain what it does. Example of list code is given below.

Contents

1. Hello World
2. Data types
3. Arithmetic operations
4. nth Fibonacci number
5. Strings
6. lists
7. Little lisper Exercise-1.3
8. Little lisper Exercise-2.5,2.6
9. Little lisper Exercise-3.1,3.2
10. Little lisper Exercise-4.1,4.2

All the programs are implemented in lisp and can be run with:

$ clisp <program-name>.lisp

1 Program 1

1.1 Lisp Code

```
;;; hello_world.lisp
;;; customary hello world program
;;; by: Kushagra Indurhya

(print "Hello World !")
```

1.2 Explanation

(print obj)- prints the object

2 Program 2

2.1 Lisp Code

```
;;; type.lisp
;;; Finding the type of an object
;;; by: Kushagra Indurhya

(setf a 10)
(setf b 10.0001)
(setf c '(1 2))
(setf d "Kushagra")

;; Takes an object and returns its type
(defun give_type_of (x)
    "returns type of x"
    (type-of x)
)

(print (give_type_of a))
(print (give_type_of b))
(print (give_type_of c))
(print (give_type_of d))
```

2.2 Explanation

Output:

(INTEGER 0 281474976710655)
SINGLE-FLOAT

CONS
(SIMPLE-BASE-STRING 8)

(setf var-name value) - creates a variable with given value
we define a function give-type-of which returns the type of the object provided to it using defun.
(defun func-name (body)) - create a function of func-name

3 Program 3

3.1 Lisp Code

```
;;; arithmetic.lisp
;;; by: Kushagra Indurhya

;;Performing arithmetic operations on user inputs using lamdas

(print "Arithematic operations")
(terpri) ;newline
(print "Enter two numbers to add : ")
(print ((lambda (x y) (+ x y)) (read) (read))) ;addition
(terpri) ;newline
(princ "Enter two numbers to subtract : ")
(print ((lambda (x y) (- x y)) (read) (read))) ;subtraction
(terpri) ;newline
(princ "Enter two numbers to multiply : ")
(print ((lambda (x y) (* x y)) (read) (read))) ;multiplication
(terpri) ;newline
(princ "Enter two numbers to divide : ")
(print ((lambda (x y) (/ x y)) (read) (read))) ;division
(terpri) ;newline
```

3.2 Explanation

Using lambdas for performing simple binary arithmetic operations in prefix(polish) notation of lisp.
(read)-takes user input

4 Program 4

4.1 Lisp Code

```
;;; fibonacci.lisp
;;; Calculating the nth fibonacci number
;;; by: Kushagra Indurhya

;;fibonacci(n) takes n as argument and returns nth fibonacci number

(defun fibonacci(n)
```

```
    "Recursive function to calculate nth number of fibonacci series"
    (cond
        ((= n 1) 0) ;if n==1 return 0
        ((= n 2) 1) ;if n==2 return 1
        (t ( + (fibonacci (- n 1)) (fibonacci (- n 2)) ) ) ; else call recurse on
             n-1 and n-2
    )
)

(print(fibonacci (read))) ; calling fibonacci() on user input
```

4.2 Explanation

Demonstrating the use of recursive function and cond operator in lisp.
cond ((condition-1)(expression) (condition-2)(expression).....)

"" - contains function documentation

5 Program 5

5.1 Lisp Code

```
;;; strings.lisp
;;; Playing with strings
;;; by: Kushagra Indurhya

;;setting variables
(setf a "Mini")
(setf b "Assignment-1")
(setf c "Lisp is cool")


(print (length c)) ; calculates length of string c
(print (subseq c 5 7)) ; prints subsequence of string c from indices 5 7
(print (concatenate 'string a b)) ; concatenating strings a and b
(print (sort (vector a b c) 'string<)); creating a vector of strings a,b & c and
                                      ;sorting it using string comparator <
(print (reverse c)); printing the reversed the alphabets of the string c
```

5.2 Explanation

Output:
12
"is"
"MiniAssignment-1"
("Assignment-1" "Lisp is cool" "Mini")
"looc si psiL"

Playing with strings and its various operations

6 Program 6

6.1 Lisp Code

```
;;; list.lisp
;;; Exploring lists
;;; by: Kushagra Indurhya

;;Defining a list for checking
(setf list '(0 1 2 3 4 5 6 7 8 9))

;; Function for calculating the length of list l
(defun len(l)
"Recursively calculates length of a list l."
    (if (null l) ; if list is empty
        0 ; return 0
            (+ 1 (len (cdr l))) ; else add 1 to the length of remaining list after
                 removing the top
))

;; Function find-nth returns the nth element of list l
(defun find-nth (n l)
  "Returns the n'th member of a list l."
  (if (null l) ;if list is empty
    nil ; return nil
        (if (= n 1) ; if list is not empty and n=1
            (first l); return first element of the list l
                (find-nth (- n 1) (cdr l)); else find the n-1th element of remaining
                      of the list after removing the top
        )
    )
)

(print (len list))
(print (find-nth 6 list))
```

6.2 Explanation

Output :
10
5
null l -> True if l is empty
if (condition) (then-do-this) (else-do-this) ->if-else in lisp
(cdr l)->takes a list l and returns l without its first element

7 Program 7

7.1 Lisp Code

```
;;; exercise-1.lisp
;;; Implementing some exercise question from chapter-1 of little lisper
;;; by: Kushagra Indurhya
;;setting variables
(setf a 'all)
(setf b 'these)
(setf c 'problems)
(setf d ())
;; Exercise 1.3
(print (cons a(cons b (cons c d)))) ;prints (ALL THESE PROBLEMS)
(print(cons a (cons (cons b d) (cons c d))));prints (ALL (THESE) PROBLEMS)
(print(cons (cons a (cons b d)) (cons c d)));prints ((ALL THESE) PROBLEMS)
(print (cons (cons a(cons b (cons c d))) d));prints ((ALL THESE PROBLEMS))
```

7.2 Explanation

Output :
(ALL THESE PROBLEMS)
(ALL (THESE) PROBLEMS)
((ALL THESE) PROBLEMS)
((ALL THESE PROBLEMS))

' / quote - in lisp skips the evaluation
cons comprises of two parts car its first elements and cdr remaining elements
(cons l1 l2) - returns a cons with l1 as its first element followed by l2

8 Program 8

8.1 Lisp Code

```
;;; exercise-2.lisp
;;; Implementing some exercise question from chapter-2 of little lisper
;;; by: Kushagra Indurhya


;;setting values
(setf l1 '(german chococate cake))
(setf l2 '(poppy seed cake))
(setf l3 '((linzer) (torte) ()))
(setf l4 '((bleu cheese) (and) (red) (wine)))
(setf l5 '(() ()))
(setf a1 'coffee)
(setf a2 'seed)
(setf a3 'poppy)

;;Exercise 2.5
```

```
(defun nonlat? (list)
"Determines whether a list of S-expressions does not contain atomic S-expression"
    (cond
        ((null list) t) ;if list is empty return true:
        (
            (not (atom (car list))) (nonlat? (cdr list)) ;car of list is not an atom
                  check for cdr
        )
        (t nil)
    )
)

(print (nonlat? l1))
(print (nonlat? ()))
(print (nonlat? l3))
(print (nonlat? l4))

;;Exercise 2.6
(defun member-cake? (list)
"Determines where a lat contains the atom \'cake "
  (if (member 'cake list) t nil) ;check if 'cake is member of list
)

(print (member-cake? l1))
(print (member-cake? l2))
(print (member-cake? l5))
```

8.2 Explanation

Output:
;Exercise 2.5
NIL
T
NIL
T
;Exercise 2.6
T
T
NIL

In nonlat? - checking if car of the list is not an atom then checking for cdr by calling nonlat? on cdr.

(member obj list) ->checks if obj is a member of list In member-cake - checking if 'cake is a member by using operator member.

9 Program 9

9.1 Lisp Code

```
;;; exercise-3.lisp
;;; Implementing some exercise question from chapter-3 of little lisper
;;; by: Kushagra Indurhya

;;Setting variables
(setf l1 '(('paella 'spanish) ('wine 'red) ('and 'beans)))
(setf l2 ())
(setf l3 '('cincinati 'chili))
(setf l4 '('texas 'hot 'chili))
(setf l5 '('soy 'sauce 'and 'tomato 'sauce))
(setf l6 '(('spanish) () ('paella)))
(setf l7 '(('and 'hot) ('but 'dogs)))
(setf a1  'chili)
(setf a2  'hot)
(setf a3  'spicy)
(setf a4  'sauce)
(setf a5  'soy)

;; Exercise 3.1
(defun seconds (l)
"Takes a lis of lats and makes a new lat consisting of the second atom from each lat
     in the list"
   (cond
     ((null l) ()) ;if list is empty return null
     (t (cons
         (car (cdr (car l))) ;making cons of second element of l and seconds of cdr
               of l
         (seconds (cdr l))
         )
     )
   )
 )
(print(seconds l1))
(print(seconds l2))
(print(seconds l7))
;; Exercise 3.2
(defun dupla (a l)
 "creates a new lat containing as many a's as there are elements in l"
   (cond
     ((null l) ()) ;if list is empty return null
     (t
       (cons a (dupla a (cdr l))) ;making cons of a and (dupla of cdr of 1)
     )
   )
)
(print (dupla a2 l4))
(print (dupla a2 l2))
(print (dupla a1 l5))
```

9.2 Explanation

Output:
;;Exercise 3.1
('SPANISH 'RED 'BEANS)
NIL
('HOT 'DOGS)

;;Exercise 3.2
(HOT HOT HOT)
NIL
(CHILI CHILI CHILI CHILI CHILI)

In seconds(l):
If list l is null then return empty list
Other cases create a cons of second element of l (cons of car of cdr of cons of l) and call seconds on cdr of l

In dupla(l):
If list l is null then return empty list
Other cases create a cons of a and (dupla a (cdr(l)))

10 Program 10

10.1 Lisp Code

```
;;; exercise-4.lisp
;;; Implementing some exercise question from chapter-3 of little lisper
;;; by: Kushagra Indurhya

;; setting up variables
(setf vec1 '(1 2))
(setf vec2 '(3 2 4))
(setf vec3 '(2 1 3))
(setf vec4 '(6 2 1))
(setf l ())
(setf zero 0)
(setf one 1)
(setf three 3)

;;defining Struct obj containing x,y
(defstruct obj x y)

;Exercise : 4.1
(defun duplicate (n obj)
"duplicates obj n times"
    (cond
        ((= zero n) ()) ;if n==0 return ()
        (t (cons obj (duplicate (- n 1) obj))) ; cons where car is obj and cdr is
             duplicating obj n-1 times
    )
```

```
)

(print(setq obj (make-obj :x 10 :y 20))) ; making a variable obj of type obj with
    values (10,20)
(print(duplicate three obj))
(print(duplicate zero obj))
(print(duplicate one vec1))

;Exercise 4.2
(defun multvec (vec)
"multiplies all elements in vec"
   (cond(
       (null vec) 1) ;if vec is empty return l
       (t (* (car vec) (multvec (cdr vec)))) ; car of vec *   (calling multvec on
            cdr of vec)
   )
)
(print (multvec vec1))
(print (multvec vec3))
(print (multvec l))
```

10.2 Explanation

Output:
;;Exercise 4.1 S(OBJ :X 10 :Y 20)

(S(OBJ :X 10 :Y 20) S(OBJ :X 10 :Y 20) S(OBJ :X 10 :Y 20))
NIL
((1 2))

;;Exercise 4.2

2
6
1

duplicate(n obj):
if n==0 return epty list
otherwise return a cons of obj and (calling duplicate with n-1 on obj) hence ducplicated n times
multvec(vec):

if vector is null return epty list
otherwise return (car of l * (call multvec on cdr of vec))