MiniAsgn1

Ananya Mantravadi - CS19B1004

March 2021

1 Program 1

1.1 Lisp Code

```lisp

;; Return the sum of three numbers

(defun add (x y z)
  "Add three numbers x, y and z"
  (+ x y z))

  (write (add 6 7 8) ) ; Prints 21
```

1.2 Explanation

The function add takes a list of elements x, y, z as arguments and returns their sum.

2 Program 2

2.1 Lisp Code

```lisp

;; Return the difference of two numbers

(defun diff (x y)
  "Subtract y from x"
  (- x y))

  (write (diff 6 7) ) ; Prints -1
```

2.2 Explanation

The function diff takes a list of elements x, y as arguments and returns their difference.

3 Program 3

3.1 Lisp Code

```

;; Return the average of four numbers

(defun average (a b c d)
   (/ ( + a b c d) 4))

(write(average 10 20 30 40)) ; Prints 25
```

3.2 Explanation

The function average takes arguments a, b, c, d and returns their average. As Lisp uses prefix notation, first it calculates the sum and then divides it by 4.

4 Program 4

4.1 Lisp Code

```

;; Return the factorial of a number greater than 0

(defun factorial (n)
        (if (= n 1)
                1
                (* n (factorial (- n 1) ))
        ))
(write (factorial 4)) ; Prints 24
```

4.2 Explanation

This is a recursive function that takes an argument which is a positive number and returns its factorial. If the argument is 1, it returns 1 as specified by the if condition. Else, it recursively calls the factorial function, multiplying n with factorial of n-1.

5 Program 5

5.1 Lisp Code

```

;; Return the n-th fibonacci number

(defun fibonacci (n)
  (cond
    ((= n 0) 0)
    ((= n 1) 1)
    (t(+ (fibonacci (- n 1))(fibonacci (- n 2))))))

(write (fibonacci 10)) ; Prints 55
```

5.2 Explanation

This is a recursive function that takes an argument which is a positive number and returns n-th fibonacci number. The first two fibonacci numbers are fixed as 0 and 1, specified by cond. Then we add n-1 and n-2 fibonnaci numbers recursively.

6 Program 6

6.1 Lisp Code

```

;; Return the coefficient of the term x^r in the binomial expansion of (1 + x)^n

(defun binomial (n  r)
  (if (or (= r 0) (= r n))
      1
    (+(binomial (- n 1) (- r 1)) binomial (- n 1) r))))

(write (binomial 4 2)) ; Prints 6
```

6.2 Explanation

This is a recursive function that takes two arguments n and r, returning the coefficient of the term x^{r} in the binomial expansion of $(x+1)^{\mathrm{n}}$. For example, B(4, 2) = 6 because $(x+1)^4 = 1 + 4x + 6x^2 + 4x^3 + x^4$

7 Program 7

7.1 Lisp Code

```

;; Function concat concatenates L2 to L1

(defun concat (L1 L2)
 (if (null L1) L2
  (cons (first L1) (concat (rest L1) L2))))

(write(concat ’(a 5 8) ’(z 9 o))) ; Prints (A 5 8 Z 9 O)
```

7.2 Explanation

This is a recursive function that takes two lists as an argument and concatenates L2 to L1. cons takes two arguments, an element and a list and returns a list with the element inserted at the first place. So this is how the recursive function works:
(concat (a 5 8) (z 9 o))
(concat (5 8) (z 9 o))
(concat (8) (z 9 o))
(concat NIL (z 9 o)), returned (z 9 o)
returned (8 z 9 o)
returned (5 8 z 9 o)
returned (a 5 8 z 9 o)

8 Program 8

8.1 Lisp Code

```

;; Returns a list containing the same elements in L except for the last one

(defun wlast (l)
 (cond
  ((null l) nil)
  ((null (cdr l)) nil)
  ((cons (first l) (wlast (rest l))))))

 (write(wlast ’(90 hi there))) ; Prints (90 HI)
```

8.2 Explanation

This is a recursive function that takes a list an argument and returns a list containing the same elements in L except for the last one. We assume that (wlast nil) and (wlast single-element-list) returns nil. So this is how the recursive function works:
wlast(90 hi there)
wlast(hi there)
wlast(there), returned NIL
returned (hi)
returned (90 hi)

9 Program 9

9.1 Lisp Code

```lisp

;; Return the n-th power of a number

(defun power (x e)
 (cond
  ((= e 0) 1)
  ((= e 1) x)
  (t(* x (power x (- e 1))))))

(write(power 2 4)) ; Prints 16
```

9.2 Explanation

This is a recursive function that takes two arguments x and e and returns the value of x^{e} by recursively multiplying x e times with itself. So this is how the recursive function works:
power(2 4)
power(2 3)
power(2 2)
power(2 1), returned 2
returned 4
returned 8
returned 16

10 Program 10

10.1 Lisp Code

```lisp

;; Return T if number is perfect, else nil

(defun perfectn (n)
  (= n (loop for i from 1 below n when (= 0 (mod n i)) sum i)))

(write(perfectn 9)) ; Prints NIL
(write(perfectn 6)) ; Prints T
```

10.2 Explanation

A perfect number is a positive integer that is the sum of its proper positive divisors excluding the number itself. This function takes a number as an argument, and checks if it is a perfect number. We loop from 1 to n-1 and add it to the sum whenever it divides the given number completely. If this sum is equal to the given number, it is perfect and returns T and if it's not, it returns NIL.