1

(defun **palindromo** (l)

(equalp l (reverse l))

)

(defun ingreso\_palindromo ()

(let (lista)

(print “Ingrese lista”)

(setq lista (read))

(if (consp lista)

(**palindromo** lista)

(pprint “no es una lista”))))

1. **CLIMA**

(Defun **clima** (a)

(cond ((< a 0) (pprint "helado"))

((and (>= a 0)(< a 10))(pprint "frio"))

((and (>= a 10)(< a 20))(pprint "templado"))

((and (>= a 20)(< a 30))(pprint "calido"))

((> a 30) (pprint "abrasador"))))

(defun **ingreso\_clima** ()

(let (temperatura)

(pprint "Ingrese temperatura: ")

(setq temperatura (read))

(if (numberp temperatura)

(**clima** temperatura)

(printf "DATO INVALIDO"))))

1. **AREA TRIANGULO**

(defun **calculaArea** (a b)

(/(\* a b) 2))

(defun **areaTriangulo** ()

(let ((base 0) (altura 0))

(pprint "Ingrese valor de la base: ")

(Setq base (read))

(pprint "Ingrese valor de la altura")

(Setq altura(read))

(if (and (numberp base) (numberp base))

(**calculaArea** base altura)

'ERROR)))

1. **CUADRADO PERFECTO:**

(defun es\_**cuadrado\_perfecto** ()

(let (valor)

(pprint "Ingrese valor a analizar: ")

(setq valor (read))

(if (integerp(truncate (sqrt valor)))

'T

'NIL)))

1. **TRIANGULOS**

(defun **clasificoTriangulos** ()

(let ((a 0) (b 0) (c 0))

(pprint "Ingrese longitud del lado 1: ")

(setq a (read))

(pprint "Ingrese longitud del lado 2: ")

(setq b (read))

(pprint "Ingrese longitud del lado 3: ")

(setq c (read))

(if (and (< a (+ b c)) (> (- b c))) /// se tiene que cumplir para todos los lados

(**calculoClasif** a b c)

(pprint "NO ES UN TRIANGULO"))))

(Defun **calculoClasif** (a b c)

(cond ((= a b c) (pprint "ES UN EQUILATERO"))

((or (= a b) (= b c) (= a c)) (pprint "ES UN ISOSCELES"))

((and (/= a b) (/= b c) (/= a c)) (pprint "ES UN ESCALENO"))))

1. **TEMPERATURAS**

(defun busca\_minima ()

(let ((temperatura 0) (mar\_list ()) (abr\_list ()))

(pprint "Ingrese temperatura a buscar: ")

(setq temperatura (read))

(pprint "Ingrese dias de marzo con su temperatura minima")

(setq mar\_list (read))

(pprint "Ingrese dais de abril con su temperatura minima")

(setq abr\_list (read))

(cond((member temperatura mar\_list)(pprint "TEMPERATURA REGISTRADA EN MARZO"))

((member temperatura abr\_list)(PPRINT "TEMPERATURA REGISTRADA EN ABRIL"))

(T (pprint"EL VALOR BUSCADO NO COINCIDE CON MARZO NI ABRIL")))))

1. **INVERTIR LISTAS**

**IZQUIERDA**

(defun izquierda ()

(let (l)

(pprint "ingrese lista: ")

(Setq l (read))

(if (consp l)

(cons(car(last a)) (butlast a))

(pprint "el valor ingresado no es una lista"))))

/////////////////////////////////////////////////////////////////////////////////////////

**DERECHA**

(defun derecha()

(let (l)

(pprint "ingrese lista: ")

(Setq l (read))

(if (consp l)

(append(cdr l)(list(car l)))

(pprint "el valor ingresado no es una lista"))))

1. **MEIDANO**

(ESTO NO ESTA RESUELTO)

(defun **calculo\_mediano** (a b c)

(setq lista (list a b c))

(cond

((and(/= a (max lista))(/= a (min lista)))

(pprint"El numero dle medio es 2"))

((and(/= b (max lista))(/= b (min lista)))

(pprint"El numero dle medio es 2"))

((and(/= b (max lista))(/= b (min lista)))

(pprint"El numero dle medio es 2"))))

(defun **ingreso** ()

(let((a)(b)(c))

(pprint"Ingrese primer valor")

(setq a(read))

(pprint"Ingrese segundo valor")

(setq b(read))

(pprint"Ingrese tercer valor")

(setq c(read))

(if(and(numberp a)(numberp b)(numberp c))

(calculo\_mediano a b c)

(pprint "NO SON VALORES NUMERICOS"))))

1. **POSICION**

(defun posicion (a b)

(if (member a b)

(-(length b)(length(member a b)))))

(defun ingreso ()

(let ((a)(b))

(pprint "Ingrese el atomo")

(setq a (read))

(pprint "Ingrese la lista")

(setq b (read))

(posicion a b)))

**10. MI SEGUNDO**

(defun **ejecutaFunc** ()

(let (a b)

(pprint "Ingrese valor atomico: ")

(setq a (Read))

(pprint "Ingrese lista: ")

(setq lista (read))

(if(and (atom a) (consp lista))

(**mi\_segundo** a lista)

(pprint "Los valores ingresados no coinciden con los solicitados"))))

(defun **mi\_segundo** (a b)

(cons (car b)(cons a (cdr b))))

**14. DISPERSION**

(defun **nivel\_rio** ()

(let ((a 0)(b 0)( c0))

(pprint "Ingrese el valor delprimer dia en CM")

(setq a (read))

(pprint "Ingrese el valor delprimer dia en CM")

(setq b (read))

(pprint "Ingrese el valor delprimer dia en CM")

(setq c (read))

(if (and(numberp a)(numberp b)(numberp c))

(dispersion a b c)

(pprint "ERROR"))

(CLASIFICACION A B C)))

(defun **dispersion** (a b c)

(-(max a b c)(min a b c )))

(defun **clasificacion** (a b c )

(cond

((< (-(max a b c)(min a b c )) 30)(pprint "DIAS PAREJOS"))

((>(-(max a b c)(min a b c )) 100)(pprint"DISPERSION GRANDE"))

(T(pprint "DIAS NORMALES"))))

15.

**A.PESO PINO**

(defun **ingreso** ()

(let (peso)

(pprint "Ingrese altura del pino")

(setq peso (read))

(if (numberp peso)

(**pesopino** peso)

(pprint "El valor no es numerico"))))

(defun **pesopino** (a)

(cond

((<= a 3) (\* a 3))

((> a 3) (+(\*(\*(- a 3) 2)100) 900))))

**B. PESO ÚTIL**

(defun esPesoUtil(a)

(if (and(>=(+(\*(\*(- a 3) 2)100) 900) 400) (<=(+(\*(\*(- a 3) 2)100) 900) 1000))

'T

'NIL))

**16.**

**PIZZAS**

(defun **cuanto** ()

(let ((a)(b))

(pprint "Ingrese precio")

(setq a (read))

(pprint"Ingrese cantidad de personas")

(setq b (read))

(if (and (numberp a)(numberp b))

(**calculo** a b)

'ERROR)))

(defun **calculo** (a b)

(/(\*(ceiling(float(/(\* b 3)8)))a)b))

**\*\*MODELO PARCIAL AZUCAR**

(defun **calculo** (a)

(print (list a(floor(float(/(/ a 4) 80))))))

(defun **ingreso** ()

(let (a) (pprint "Ingrese presupuesto")

(setq p (read))

(if (numberp p)

(calculo p)

'ERROR)

(calculo\_cafe p)))

**\*\*CAFÉ**

(defun **calculo\_cafe** (a)

(if(>=(/(float(- a(/ a 4)))1500) 2) 'T ))