**1-**

**a)**  
;; Define uma lista com elementos traduzidos

(setq minha-lista '(A b C x y Z))

;; Imprime o primeiro elemento

(format t "Primeiro elemento: ~a~%" (car minha-lista))

;; Imprime o último elemento

(format t "Último elemento: ~a~%" (car (last minha-lista)))

**b)**

;;Cria a lista de divisores dos primeiros divisores de 24

(setq divisores '(1 2 3 4 6 8 12 24))

;;Cria uma nova lista chamada "nova-lista" (inicialmente vazia)

(setq nova-lista '())

;;Realizar o procedimento de remoção e inserção três vezes

(dotimes (i 3)

;;Remove o primeiro elemento da lista "divisores"

(let ((primeiro-elemento (car divisores)))

(setq divisores (cdr divisores)) ; Atualizar a lista de divisores sem o primeiro elemento

;;Adiciona o primeiro elemento à lista "nova-lista"

(push primeiro-elemento nova-lista)))

;;Imprime as duas listas

(format t "Lista 'divisores': ~a~%" divisores)

(format t "Lista 'nova-lista': ~a~%" nova-lista)

**c)**

;; Cria uma lista com as informações de notas

(setq notas-aluno '((PP . 9.0) (ES . 8.5) (MPFC . 7.0) (IHC . 8.0)))

;; Realiza a busca pela disciplina "PP" na lista

(setq disciplina-buscada 'PP)

(setq nota-encontrada nil)

;; Itera sobre a lista de notas para encontrar a disciplina "PP"

(dolist (disciplina-nota notas-aluno)

(let ((disciplina (car disciplina-nota))

(nota (cdr disciplina-nota)))

(if (equal disciplina disciplina-buscada)

(setq nota-encontrada nota))))

;; Imprime a nota do aluno na disciplina "PP" na tela

(if nota-encontrada

(format t "Nota do aluno em PP: ~a~%" nota-encontrada)

(format t "Disciplina 'PP' não encontrada nas notas do aluno.~%"))

**2-**

**A)**

(defun rotacao-direita (lista)

"Realiza uma rotação da lista para a direita, movendo o último elemento para a primeira posição."

(if (endp lista) ; Verifica se a lista está vazia

lista ; Se a lista estiver vazia, não é possível realizar a rotação

(let\* ((ultimo-elemento (car (last lista))) ; Obtém o último elemento

(lista-sem-ultimo (butlast lista))) ; Remove o último elemento

(cons ultimo-elemento lista-sem-ultimo)))) ; Adiciona o último elemento na primeira posição

;; Exemplo de uso:

(format t "Lista original: ~a~%" '(1 2 3 4 5))

(format t "Rotação para a direita: ~a~%" (rotacao-direita '(1 2 3 4 5)))

**B)**

(defun palindromizar (lista)

"Cria um palíndromo a partir de uma lista, dobrando seu tamanho."

;;Inverte a lista

(setq lista-invertida (reverse lista))

;; Concatena a lista original com a lista invertida

(setq palindromo (append lista lista-invertida))

;; Retorna o palíndromo

palindromo)

;; Exemplo de uso:

(format t "Lista original: ~a~%" '(ARARA))

(format t "Palíndromo: ~a~%" (palindromizar '(ARARA)))

**c)**

;; Procedimento para converter graus Fahrenheit para Celsius

(defun fahrenheit-para-celsius (f)

"Converte graus Fahrenheit para Celsius."

;; Aplicar a fórmula correta

(let ((c (/ (\* (- f 32) 5) 9))) ; Definir uma variável local "c" com o valor da temperatura em Celsius

c))

;; Procedimento para converter graus Fahrenheit para Celsius

(defun fahrenheit-para-celsius (f)

"Converte graus Fahrenheit para Celsius."

;; Aplicar a fórmula correta

(let ((c (/ (\* (- f 32) 5) 9))) ; Definir uma variável local "c" com o valor da temperatura em Celsius

c)) ; Retornar o valor de "c"

;; Procedimento para converter graus Celsius para Fahrenheit

(defun celsius-para-fahrenheit (c)

"Converte graus Celsius para Fahrenheit."

;; Aplicar a fórmula correta

(let ((f (+ (\* c 9/5) 32)))

f))

;; Exemplos de uso:

(format t "32 graus Fahrenheit é igual a ~a graus Celsius.~%" (fahrenheit-para-celsius 32))

(format t "0 graus Celsius é igual a ~a graus Fahrenheit.~%" (celsius-para-fahrenheit 0))

**3)**

**a)**

(defun verifica-segundo-maior-que-5 (lista)

"Verifica se o segundo número da lista é estritamente maior que 5."

;; Passo 1: Verificar se a lista tem pelo menos dois elementos

(if (>= (length lista) 2) ; Verifica se a lista tem pelo menos 2 elementos

(let ((segundo (nth 1 lista))) ; Obtém o segundo elemento da lista

;; Passo 2: Verificar se o segundo número é estritamente maior que 5

(if (> segundo 5)

(format t "true") ; Se for verdade, imprime "true"

(format t "false"))) ; Caso contrário, imprime "false"

(format t "A lista não possui pelo menos dois elementos"))) ; Se a lista não tiver dois elementos, informa

;; Exemplo de uso:

(format t "Resultado: ")

(verifica-segundo-maior-que-5 '(3 1 5 0))

**b) \*\*\*\***

(defun operacao-com-parametros (x operacao)

"Executa uma operação com base no valor da string operacao e imprime o resultado."

(cond

((string= operacao "somar")

;; Executa a operação "somar" e imprime o resultado

(let ((resultado (+ (\* 10 x) 100)))

(format t "Resultado da operação 'somar': ~a~%" resultado)))

(t

;; Executa a operação padrão e imprime o resultado

(let ((resultado (- x (/ x 5) (/ x 10) (round (/ x 3.0)))))

(format t "Resultado da operação padrão: ~a~%" resultado)))))

;; Exemplo de uso:

(format t "Digite um número inteiro: ")

(setq numero (read))

(format t "Digite 'somar' ou qualquer outra palavra: ")

(setq operacao (read-line))

(operacao-com-parametros numero operacao)

**4) \*\*\*\***

(defun calcular-pa (a1 r n)

"Calcula e imprime os primeiros n termos de uma Progressão Aritmética."

(format t "Os primeiros ~a termos da PA com a1 = ~a e r = ~a são:~%" n a1 r)

(loop for i from 1 to n do

(let ((termo (+ a1 (\* (1- i) r))))

(format t "~a " termo)))

(format t "~%") ; Imprimir uma nova linha no final

)

;; Solicitar entrada do usuário

(format t "Digite o primeiro termo (a1): ")

(setq a1 (read))

(format t "Digite a razão (r): ")

(setq r (read))

(format t "Digite a quantidade de termos (n): ")

(setq n (read))

;; Chamar o procedimento para calcular e imprimir a PA

(calcular-pa a1 r n)

b)?

**5)**

(defun SEPARAR-ORGANIZAR (lista)

"Imprime todos os átomos da lista original."

;; Verificar se a lista está vazia

(if (null lista)

nil

;; Se não estiver vazia, processar cada elemento

(progn

;; Processar o primeiro elemento da lista

(if (atom (car lista))

(format t "~a " (car lista)) ; Imprimir átomo

(SEPARAR-ORGANIZAR (car lista))) ; Recursivamente processar sublista

;; Processar o restante da lista

(SEPARAR-ORGANIZAR (cdr lista)))))

;; Exemplo de uso:

(format t "Resultado: ")

(SEPARAR-ORGANIZAR '(a (a (a (a b))) (((a b) b) b) b))