Preguntas de examen

- Diferencia entre paradigma hibrido y puro (totalmente funcional o no?)
LISP: Hibrido
Scheme: funcional puro
Prolog: logico puro
SmallTalk: Obj puro

- Diferencia entre imperativo y declarativo

- Que es LISP Listener?
(PRINT (EVAL(READ)))

- Lista impropia ¿Como se crea?

- Lista Propia

- Funciones destructivas y no-destructivas

- Las variables locales son de ambito lexico

- Las variables globales son de ambito dinamico

Funciones en LISP

Nota:
Las funciones se escriben de la forma(funcion parametro)
Si son funciones dentro de funciones es: (funcion (funcion parametro))

FUNCION	DESCRIPCION	EJEMPLO
`(quote obj) o '(obj)`	Evita que un objeto se evalue. Es la forma de poder pasar una lista como datos	`(1 2 3 4) = ERROR` `'(1 2 3) = (1 2 3)`
	ARITMETICA	
`(+ a b ...)`	Suma variables	
`(- a b ...)`	Resta	
`(* a b ...)`	Multiplicacion	
`(/ a b ...)`	Division	
`(truncate <expr>)`	Trunca una expresion numerica tendiendo a 0	`(truncate 3.14) = 3` `0.14000000000000012`
`(round <expr>)`	Redondea hacia el entero positivo mas cercano	`(round 3.723) = 4` `-0.27700000000000014`
`(ceiling <expr>)`	Redondea hacia arriba	`(ceiling 5.01) = 6`
`(floor <expr>)`	Redondea hacia abajo	`(floor 8.9) = 8`
`(float <expr>)`	Convierte un entero en un numero de coma flotante.	`(float 8) = 8.0`
`(rational <expr>)`	Convierte un numero real en racional.	`(rational 2.5) = 5/2`
`(mod <expr> <expr>)`	Devuelve el remanente de la división entera de dos números.	`(mod 7 2) = 1`
`(abs <expr>)`	Devuelve el valor absoluto de una expresión.	`(abs -8) = 8`
`(signum <expr>)`	Permite obtener el signo de una expresión. Devuelve 0 para el 0, 1 para los positivos y -1 para los negativos.	`(signum -8) = -1`
`(max num num ...)`	Devuelve el mayor de una lista de numeros.	`(max 2 5 3 1) = 5`
`(min num num ...)`	Devuelve el menor de una lista de números.	`(min 2 8 9 4) = 2`
`(gcd num num...)`	**Máximo Común Divisor** de una lista de números. Si no recibe argumentos, devuelve 0. Si tiene un solo argumento, devuelve el mismo argumento.	`(gcd 12 34 56) = 2`
`(LCM NUM NUM...)`	**Mínimo Común Múltiplo** de una lista de números. Si no recibe argumentos, devuelve error. Si tiene un solo argumento, devuelve el mismo argumento. Si el resultado es mayor que el limite de los enteros, devuelve un error.	`(lcm 12 34 56) = 2856`
	FUNCIONES MATEMATICAS	

(SQRT NUM)	Raiz cuadrada de un numero	(SQRT 16) = 4.0 (SQRT -4) = #c(0.0 2,0)
(EXPT NUM_A NUM_B)	Exponencial	(EXPT 2 3) = 8
	Estas funciones *no modifican* el valor de los argumentos pasados como parámetros.	
(INCF Var [DELTA])	Incrementa en una cantidad DELTA un valor Var, por defecto es 1.	(INCF 5 10) = 15
(DECF Var [DELTA])	Decrementa en una cantidad DELTA, una variable Var, por defecto es 1.	(DECF 5 10) = -5
	Estas funciones *modifican* el valor de los argumentos pasados como parámetros.	
	PREDICADOS DE COMPARACION **Def: funciones booleanas que devuelven solo T o NIL**	
(= NUM NUM …)	Igual que	(= 3 3.0) ==> T
(/= NUM NUM…)	Desigual	(/= 3 3.0) ==> NIL
(< NUM NUM …)	Menor que	(< 3 5 8) ==> T (< 3 5 4) ==> NIL
(<= NUM NUM …)	Menor o igual que	(<= 5 5 8) ==> T (<= 9 9 4) ==> NIL
(> NUM NUM …)	Mayor que	(> 8 5 3) ==> T (> 8 3 5) ==> NIL
(>= NUM NUM …)	Mayor o igual que	(>= 9 7 7) ==> T (>= 8 9 8) ==> NIL
	PREDICADOS NUMERICOS **Def: se utilian para verificar exclusivamente argumentos numéricos. Aceptan 1 solo argumento.**	
(NUMBERP OBJ)	Verifica si el tipo de objeto es numérico. Devuelve T si el objeto es un número. El argumento puede ser de cualquier tipo.	(NUMBERP 7) ==> T
(ODDP ENTERO)	Verifica un argumento, que debe ser entero, y devuelve cierto si el entero es impar.	(ODDP -7) ==> T
(EVENP ENTERO)	verifica un argumento, que debe ser entero, y devuelve cierto si el entero es par	(EVENP 8) ==> T
(INTEGERP OBJ)	verifica si el argumento es un numero entero. Devuelve true si el argumento es un entero. El argumento puede ser de cualquier tipo	(INTEGERP 7) ==> T
(ZEROP NUM)	verifica un argumento, que debe ser numérico, y devuelve cierto si el número es cero.	(ZEROP 0.0) ==> T
	FUNCIONES CON UN COMPORTAMIENTO DISTINTO AL ESPERADO	
(REVERSE ELEMENTO)	La funcion reverse funciona tanto como para listas como para elementos atómicos de tipo string, siendo este el unico elemento de tipo atómico al que es aplicable. Cumple con la funcion de invertir el orden de los elementos devolviendo una lista en caso de encontrarse al menos uno que cumpla con la condicion de ser de tipo CONS y en caso del string devuelve unicamente este invertido.	(SETQ LISTA '(120 34 4 0 99)) (REVERSE LISTA) (REVERSE "STRING")
(EQUAL ELEMENTO) (EQ ELEMENTO)	: La función `eq` compara dos objetos y devuelve `t` si son el mismo objeto (es decir, si tienen la misma dirección de memoria) y `nil` en caso contrario. Por otro lado, `equal` compara dos objetos estructuralmente, es decir, si dos objetos tienen el mismo contenido, pero no son el mismo objeto, `equal` devuelve `t`. Por lo tanto, en algunos casos, `eq` y `equal` pueden dar resultados inesperados, especialmente cuando se trabaja con objetos complejos como listas y estructuras.	(SETQ LISTA1 '(1 2 3)) (SETQ LISTA2 '(1 2 3)) (SETQ LISTA3 LISTA1) (EQUAL LISTA1 LISTA2) Devuelve T, ya que las listas tienen los mismos elementos (EQ LISTA1 LISTA2) Devuelve NIL, ya que son dos objetos diferentes (EQUAL LISTA1 LISTA3) Devuelve T, ya que LISTA3 es una referencia a LISTA1 (EQ LISTA1 LISTA3) Devuelve T, ya que LISTA3 y LISTA1 son el mismo objeto
(EQL ELEMENTO) (EQUALP ELEMENTO)	Eql evalua si x e y representan el mismo valor, no conviene utilizarlo con strings, solo numeros del mismo tipo Equalp evalua si x e y representan el mismo valor, puede aplicarse a strings y es case insensitive, por lo tanto no diferencia mayusculas de minusculas	(EQL 5 5.0) -> NIL (EQL 5 5) -> NIL (SETQ A 'WORD) (SETQ B 'WORD) (EQUALP A B) -> TRUE
	OPERADORES LOGICOS	
AND	Evalua los argumentos en orden, si cualquiera de los argumentos evalua NIL se detiene la ejecución y devuelve NIL	(AND (< 2 5) (> 7 4) (* 2 5)) -> 10

OR	Evalua los argumentos en orden, si cualquiera de los argumentos evalua NO-NIL se detiene la ejecución y devuelve T	(SETQ X 10) (OR (< 0 X) (DECF X))-> T (OR (> 0 X) (DECF X)) -> 9
NOT	Evalua un unico argumento, esta función devuelve T o NIL	(SETQ X '(A B C)) (NOT (ATOM X)) -> T
	PREDICADOS SOBRE TIPOS DE IGUALDAD	
(ATOM OBJETO)	Devuelve true si no es una construccion cons	
(CONSP OBJETO)	Devuelve true si es un objeto cons	
(LISTP OBJETO)	Devuelve true si es un objeto cons o nil	
(NULL OBJETO)	Devuelve true si el objeto es NIL	
(TYPEP OBJ TIPOESPECIF)	Devuelve true si el objeto pertenece al tipo especificado	
	OPERACIONES SOBRE LISTAS	
(LAST LISTA)	Devuelve la ultima estructura cons de la lista	(LAST '(A B C))-> (C)
(ELT LISTA INDICE)	Devuelve el elemento de la lista que esta en la posicion del indice, el indice empieza en 0	(ELT '(A(BC)) 1) -> (BC)
(nth indice lista)	igual que ELT	
	FUNCIONES DE CONSTRUCCION DE LISTAS	
(CONS SEXP SEXP)	Crea una nueva estructura cons, acepta listas o atomos, si la segunda expresion es atomo crea una lista impropia	(CONS 'A '(B C D))-> (A B C D)
(LIST SEXP SEXP)	Devuelve una lista formada por los elementos pasados como argumentos, puede ser cualquier cantidad de argumentos, pueden ser listas o atomos, si embargo solo crea listas propias	(LIST 'A 'B 'C) -> (A B C)
(APPEND LISTA LISTA ... LISTA SEXP)	Crea una nueva lista concatenando dos o mas listas, solo acepta listas excepto en el ultimo elemento que puede ser lista o atomo, no crea listas impropias	(APPEND '(A) '(B) '(C)) -> (A B C)
	SECUENCIA DE ACCIONES	
(PROGN ... ...)	Permite crear un bloque de sentencias que serán evaluadas secuencialmente, devuelve la ultima forma evaluada	(PROGN (SETQ A 23) (SETQ B 35) (SETQ C(*A B)))->805
	ESTRUCTURAS CONDICIONALES	
(IF TEST [THEN] [ELSE])	Verifica y ramifica: Verifica: devuelve NO-NIL o NIL Ramifica: es el comando a ejectuarse basándose en el resultado de la verificación	(SETQ OBJ 5) (IF (NUMBERP OBJ)"ES UN NUMERO" "NO ES UN NUMERO") -> ES UN NUMERO
(COND(TEST1 FORMA11 FORMA12... FORMA1N))	Permite expresar varias condiciones, ejecutándose la primera que se cumple, la cabeza de la lista se evalua secuencialmente hasta encontrar una que devuelva un valor NO-NIL, se ejecuta y sale de la forma cond	(SETQ A 4) (COND ((NUMBERP A) "ESTO ES UN NUMERO") (T "ESTO NO ES UN NUMERO"))-> ESTO ES UN NUMERO
(WHEN TEST FORMA1... FORMAN)	Es equivalente al if sin la forma else, si el test es verdadero se evaluan todas las formas restantes	(SETQ A 4) (WHEN(EQUAL A 4)(PRINT(* AA))(PRINT"CIERTO"))-> 16
(UNLESS TEST FORMA1...FORMAN)	Evaluara las formas siempre que el resultado de la evaluación del test sea NIL	(UNLESS(EQUAL A 4)(PRINT(* AA))(PRINT"CIERTO"))->16
(CASE KEYFORM (LIST {FORMA}))	Permite realicar ciertas acciones cuando una determinada forma keyform tome ciertos valores, expresados a traves de una lista	(SETQ MES 'JUN) (CASE MES ((ENE MAR MAY JUL AGO OCT DIC)31) ((ABR JUN SEP NOV)30) (FEB(IF(ZEROP(MOD AÑO 4))29 28)))->30)
(TYPECASE KEYFORM (TYPE {FORMA}))	Permi realizar ciertas acciones siempre que un determinado objeto sea de la clase indicada	(SETQ X 2) (TYPECASE X (STRING "ES UN STRING") (INTEGER "ES UN ENTERO") (SYMBOL(PRINT'(ES UN SIMBOLO))) (OTHERWISE(PRINT"OPERADOR")(PRINT"DESCONOCIDO"))))->"ES UN ENTERO"
(LOOP {FORMA})	Expresa una construccion iterativa que cicla continuamente, la forma de indicar que el ciclo termine es con el return	(SETQ X 1) (LOOP (PRINT X) (SETQ X(+ X 1)) (IF(= X 20)(RETURN)))-> IMPRIME 1 HASTA EL 19 Y SALE CON NIL
(DO({PARAM}/{PARAM VALOR}/{PARAM VALOR INCRE}) (TEST-EVALUCIÓN {FORMAS}*){FORMAS}*)	Tiene 3 partes: lista de param, test de final y cuerpo, la lista de param liga las variables a sus valores iniciales, y en cada ciclo se asignan nuevos Primero se actica la ligadura de los param y luego evalua el test de final y si no se cumple, se ejecuta el cuerpo	

(DOTIMES(VAR FORMA-LIMITE-SUPERIOR [FORMA-RESULTADO]))	Permite escribir procedimientos sencillos con iteración controlada por un contador Cuando comienza el ciclo se evalua la form limite superior, produciendo un valor n,empezando desde 0 hasta n-1 Por cada valor de la variable, se ejecuta el cuerpo y al final, la ligadura con la variable se elimna y se ejecuta la forma resultado dando valor al DOTIMES	
(DOLIST(VAR L [RESULTADO]) S1,S2...SN)	Asigna a var el primer elem de la lista, evalua s1..sn para cada valor de var, si no tiene mas elementos devuelve resultado, sino asigna a var el siguiente elemento e itera el proceso, si no hay valor de resultado finaliza con nil	