

**UNIVERSIDAD ANDRÉS BELLO**

**FACULTAD DE INGENIERÍA**

**INGENIERÍA CIVIL INFORMÁTICA**

**Trabajo 2 – Introducción a LISP y PROLOG**

**GUSTAVO ANDRÉS VÉLIZ LÓPEZ**

**SANTIAGO-CHILE**

**ABRIL, 2017**

**INTRODUCCIÓN**

Para la realización de la Tarea 2 es necesario saber que es LISP y PROLOG. A continuación se dará una breve descripción de cada uno.

LISP

Es un lenguaje de programación del tipo multiparadigma originado en el Instituto Tecnológico de Massachusetts (MIT) en el año 1958, convirtiéndose en el segundo lenguaje de alto nivel que sigue siendo usado actualmente.

La intención original de este lenguaje era tener un uso de notación matemática para programas de computadora pero no tardo en convertirse en uno de los lenguajes favoritos para la investigación en el área de la Inteligencia Artificial.

El nombre de LISP viene de “LISt Processing”, dado al importante uso de la estructuras de datos “List” (la cual también compone el código fuente del lenguaje).

La sintaxis de LISP está basada en las *expresiones S* (listas entre paréntesis), además el uso de funciones sigue el orden de: (*función argumento argumento)*. El mismo caso se aplica para los operadores como sumas y restas.

PROLOG

Es un lenguaje de programación usado para programar artefactos electrónicos usando el paradigma lógico con técnicas de producción final interpretada. El origen de su nombre viene del francés “*PROgrammation en LOGique*” (Programación Lógica), se ideó al principio del año 1972 en la Universidad de Aix-Marseille I y está asociado a los trabajos de inteligencia artificial y lingüística computacional.

La programación en PROLOG consiste en dos tipos de cláusulas llamas *reglas* y *hechos*. Los hechos a diferencia de las reglas, serán siempre verdaderas. Las reglas son del tipo “Cabeza :- Cuerpo.”, y es leído como “La cabeza es vedad si el cuerpo es verdad”. El cuerpo de estas reglas consisten en llamadas a predicados, estos son llamados los *objetivos* de las reglas.

**DESCRIPCIÓN DEL PROBLEMA**

Para esta tarea se le pide al alumno la realización de funciones básicas en la mayoría de los lenguajes, de modo que este tendrá que realizar la implementación de los respectivos lenguajes para luego trabajar con ellos.

Para el uso de LISP al alumno se le exige la realización de las siguientes funciones:

1. Una función cubo que tome un número “n” y devuelva su valor al cubo.
2. Una función factorial que tome un número “n” y devuelva su valor factorial.
3. Tres funciones que tomen una lista, y que devuelvan el 1er elemento, el segundo y el tercero.
4. Una función que devuelva el promedio del cuadrado de dos números.
5. Una función que calcule Ax2+Bx+C.
6. Una función que reciba una lista y la rote un elemento hacia la izquierda.

Para PROLOG se pide la realización de las siguientes funciones dado al uso del lenguaje:

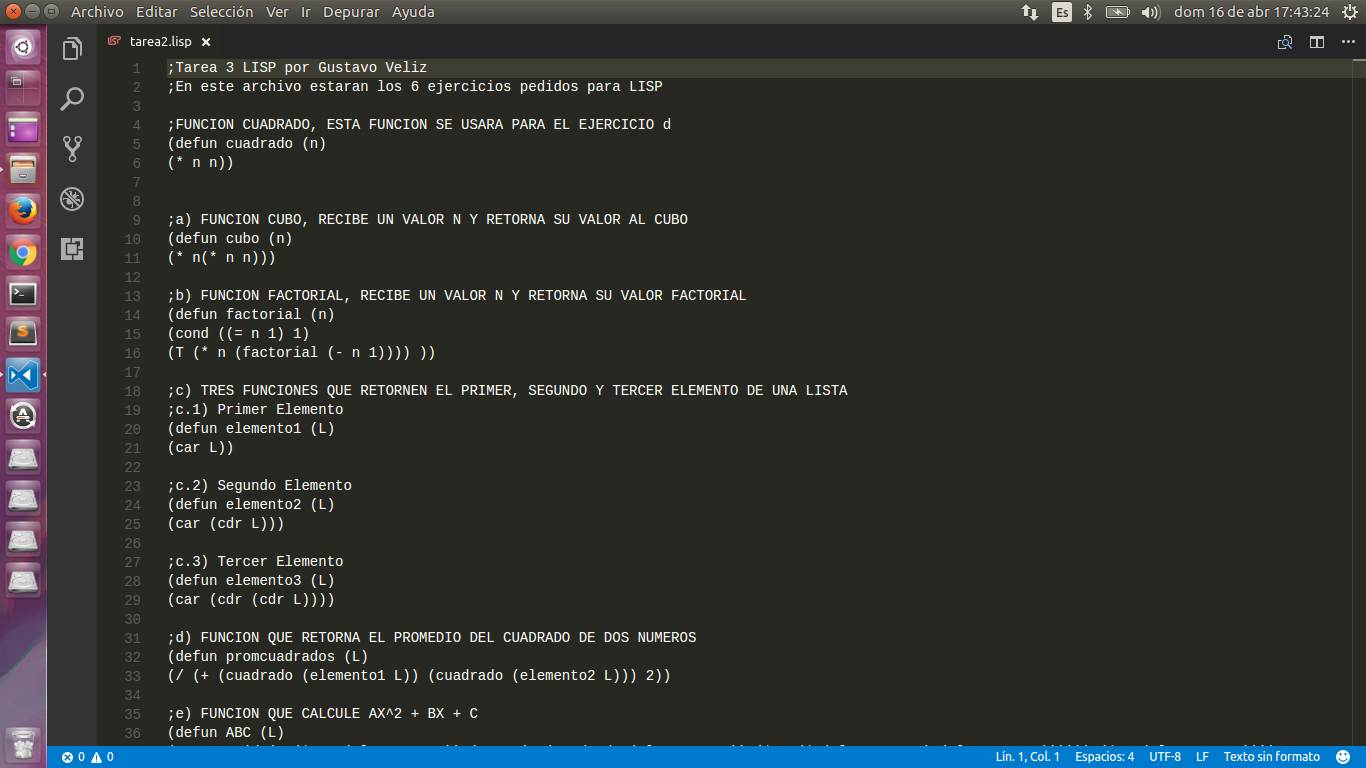
1. Una función que calcule la longitud de una lista.
2. Una función que entregue el número más grande dentro de un árbol.
3. Una función que verifique si un elemento está o no en una lista.

Además de realizar las funciones ya descritas, el alumno debe explicar en detalle su código para demostrar el conocimiento de los lenguajes requeridos.

**DESARROLLO**

Los códigos a continuación, ceñidos al respectivo lenguaje, serán explicados en detalle de modo que una persona con conocimientos básicos de informática pueda entender lo que realiza cada línea de estos.

LISP



Código 1.- LISP

Cabe destacar que todas las líneas que comienzan con “*;”* son comentarios del código, “*L”* será usado como abreviación de “*Línea”*.

**Función cuadrado**: Calcula el cuadrado del número ingresado.

L5: La sintaxis de LISP pide el uso de paréntesis para la definición de funciones y operaciones, por lo que esta línea abre el paréntesis para definir la función, “*defun*”, con el nombre “Cuadrado”, luego nuevamente entre paréntesis se indica el tipo de dato que recibirá la función, en este caso “N” para un tipo de dato numérico.

L6: Aquí se define la operación que realizara nuestra función, nuevamente se encuentra entre paréntesis y sigue el orden de *operador argumento argumento,* en este caso, es una multiplicación de un número N por sí mismo, obteniendo el cuadrado de este, por último se cierra el paréntesis que se abrió al inicio.

**Función Cubo**: Recibe un valor numérico y retorna su valor al cubo.

L10: se define la función, se le asigna el nombre “Cubo” y un tipo de dato “Numérico” como argumento.

L11: En esta línea se ve el uso de un operador dentro de un operador, de adentro hacia afuera, se realiza una multiplicación de N por N, luego al ser retornado este valor, se vuelve el segundo argumento de la operación “\*” que lo envuelve, obteniendo así el cubo de N.

**Función Factorial**: Calcula el factorial de un valor N.

L14: Aquí se define la función con el nombre “Factorial” y recibe como argumento un tipo de dato numérico.

L15: Esta línea es una condición, indica que si N es igual a 1, la función debe retornar el valor 1.

L16: T es equivalente a “*else* o *sino”,* posterior a eso viene lo que se debe realizar en caso de que la condición de la línea 15 no se cumple, en este caso debe multiplicar el valor actual por el resultado de la función factorial con N-1 como argumento (recursividad).

**Funciones que retornan primera, segunda y tercera posición de una lista.**

**Primer Elemento:**

L20: Se define el nombre de la función “elemento1” y se indica el tipo de dato que recibirá como argumento, en este caso es un dato tipo *Lista*.

L21: Se utiliza la función (que ya está integrada en LISP) llamada *car* y se le entrega la Lista antes mencionada, esta función retorna el primer elemento de la Lista.

**Segundo Elemento:**

L24: Se define el nombre de la función “elemento2” y se indica el tipo de dato que recibirá como argumento, en este caso es un dato tipo *Lista*.

L25: Aquí se introduce la función CDR, que obtiene todo lo que este después del primer elemento de la lista, por lo tanto, en esta línea se retorna esta “*cola”* de la Lista y se pide su primer elemento, obteniendo así, el segundo elemento de la Lista original.

**Tercer Elemento:**

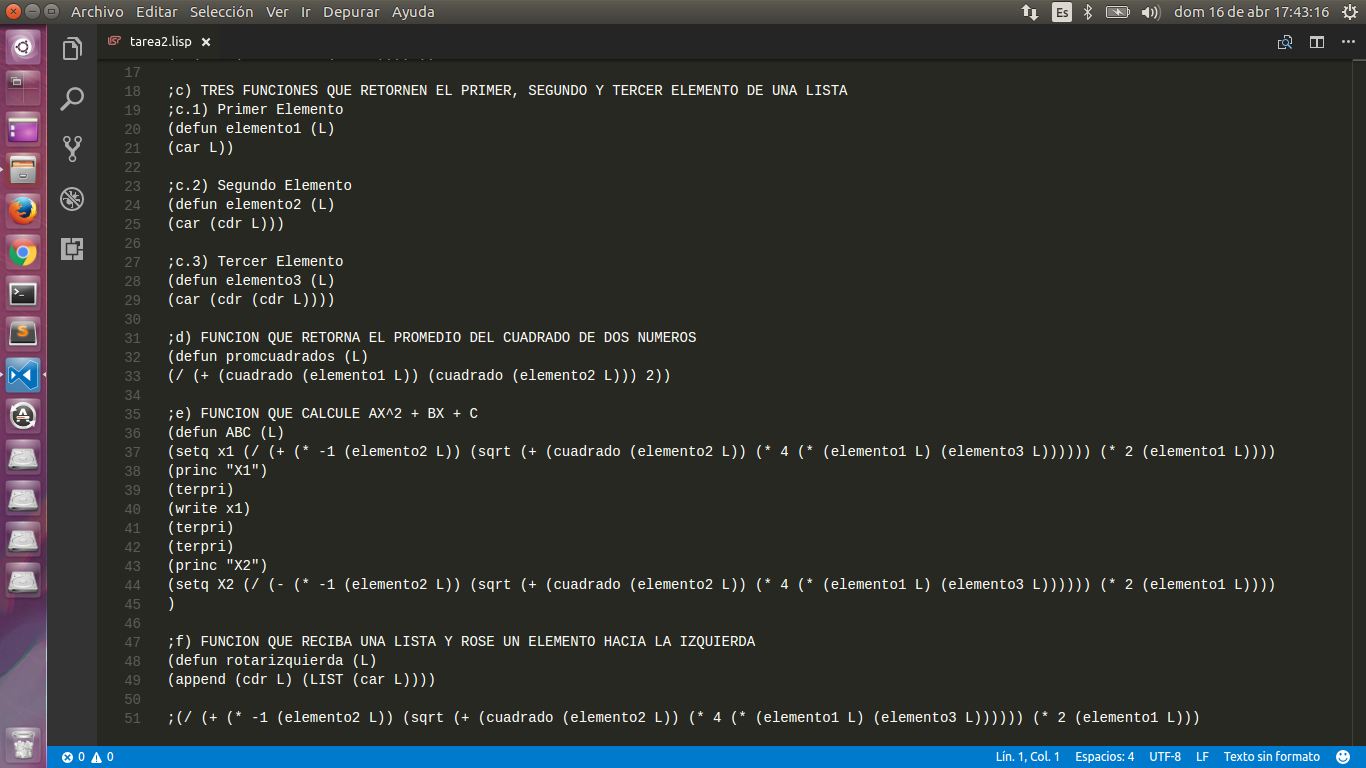
L28: Se define el nombre de la función “elemento3” y se indica el tipo de dato que recibirá como argumento, en este caso es un dato tipo *Lista*.

L29: En esta línea se pide la cola de la Lista, luego se pide nuevamente la cola de esa Lista nueva para finalmente obtener el cabezal de ella, obteniendo así el tercer elemento de la Lista original.

**Función Promedios Cuadrados:** retorna el promedio de los cuadrados de dos elementos.

L32: Se define la función con el nombre “PromCuadrados” y se le da como argumento un dato tipo Lista.

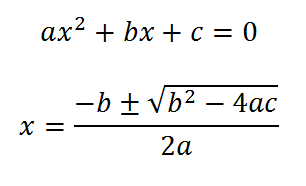
L33: Para este problema se usa las funciones “cuadrado”, “elemento1” y “elemento2” anteriormente definidas, se realiza la suma dándole como argumentos la función cuadrado que a su vez tienen como argumentos el elemento 1 y 2 de la Lista L (usando las funciones anteriormente mencionadas), finalmente el resultado de esto es divido en 2 para promediar.



Código 2.- LISP

**Función que calcule Ax^2 + Bx + C**: Calcula los valores de 0 para solucionar la ecuación dada.

L36: Se define la función con el nombre “ABC” y se le entrega Lista como argumento.

L37: En este elemento se replica la fórmula, para ello se utilizan las funciones “elemento1”, “elemento2”, “elemento3” y “cuadrado” definidas anteriormente, y las funciones “setq” que asigna a una variable, en este caso “x1”, el resultado obtenido de la operación, y “sqrt” que es para obtener la raíz cuadrada del argumento que se le entregue, estas funciones ya están integradas en LISP.

L38: La función “princ” me permite imprimir lo que se le dé como argumento. En este caso imprime “X1”, cabe mencionar que esto no es una variable, solo un mensaje.

L39: La función “terpri” permite imprimir una nueva línea. Esta función fue usada para ordenar la impresión de los resultados.

L40: La función “Write” me permite imprimir el valor del argumento que se le entrega. En este caso, imprime el valor resultante de “x1”.

L41, L42: Al igual que la Línea 39, se utiliza la función “terpri” para ordenar la salida de resultados.

L43: Al igual que la Línea 38 se imprime un mensaje, en este caso “X2”.

L44: Al igual que la línea 37, se asigna el resultado de la fórmula a la variable X2, cabe destacar que esta ecuación es con una resta al inicio al contrario de la suma del a L37. Aprovechando que LISP, imprime por si el último resultado, no se tuvo que usar otro “princ”.

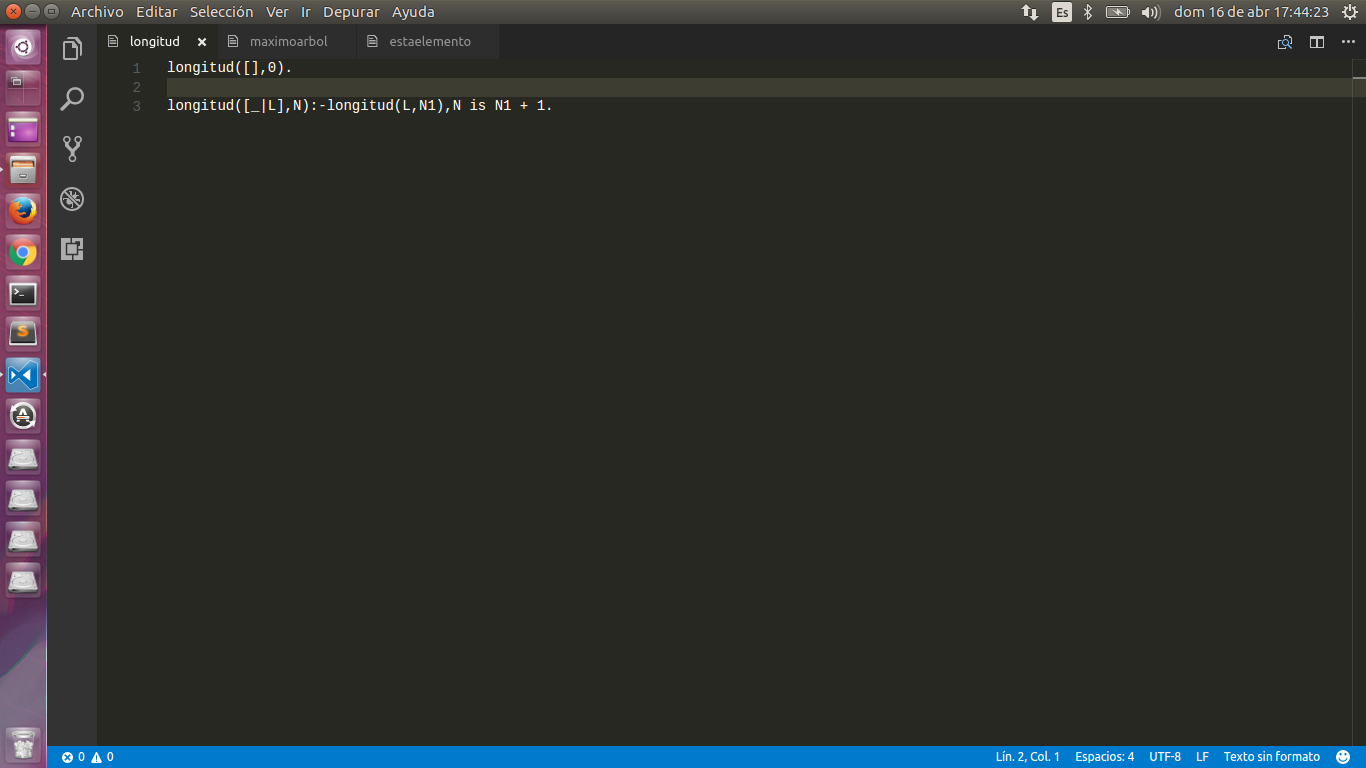
L45: Se cierra el paréntesis de toda la función.

**Función que rota un elemento hacia la izquierda de una Lista.**

L48: Se define el nombre de la función “rotarizquierda” y el tipo de dato que recibe, en este caso, Lista.

L49: “append” es una función que une dos elementos, en este caso une la cola de la lista (función “cdr”) a la cabeza de la lista (función “car), lo que nos da como resultado una rotación a la izquierda de esta. Ej: ABCD -> BCDA.

PROLOG

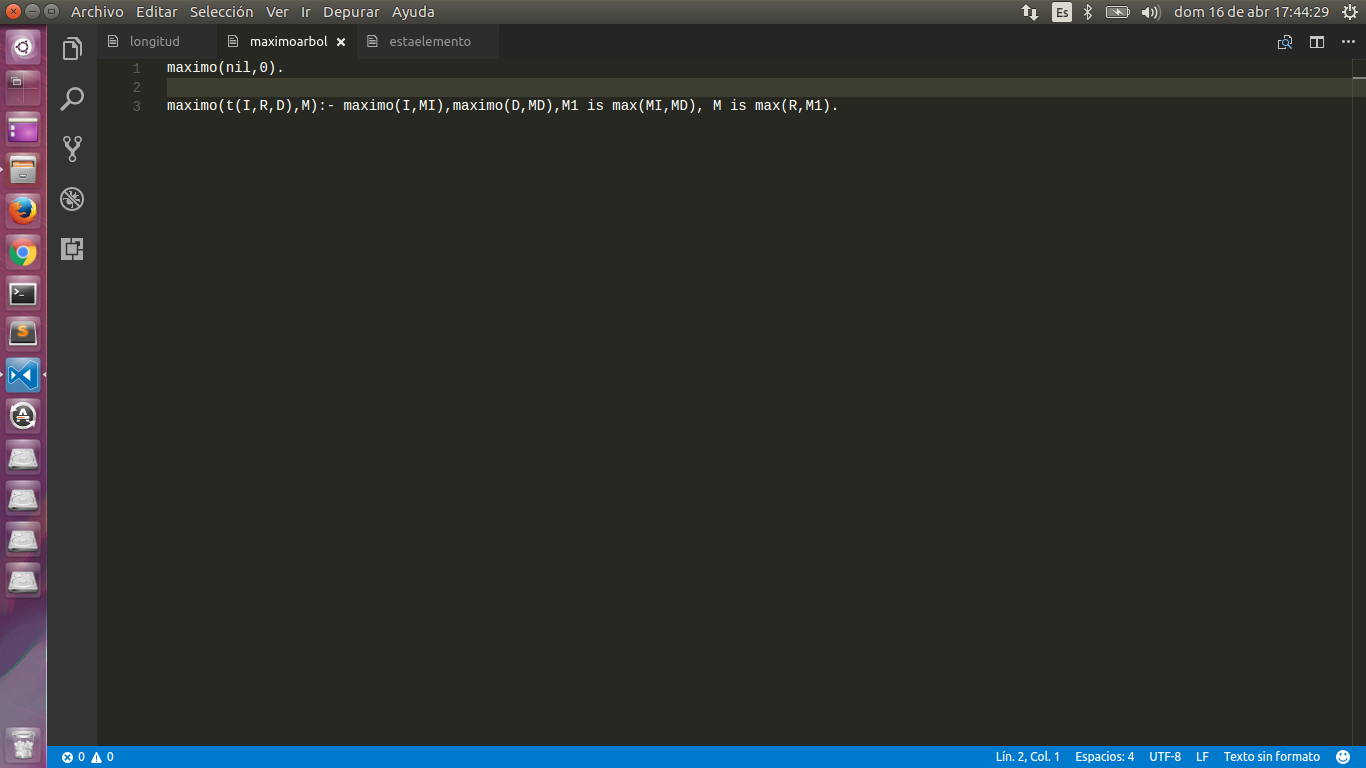


Código 3.- PROLOG

**Función Longitud**: Calcula la longitud de una Lista.

L1: Esta línea es un *hecho*, significa que cuando se llame a la función, y la lista este vacía, esta debe retornar un 0.

L3: Esta línea es una *regla*, y usa recursividad para obtener el largo de la lista. Empieza a eliminar el elemento que este en la cabeza de la lista hasta cumplir el hecho de la L1, luego debido a la recursión, empieza a realizar la suma de N1’s, obteniendo finalmente el largo de la Lista en la variable N.

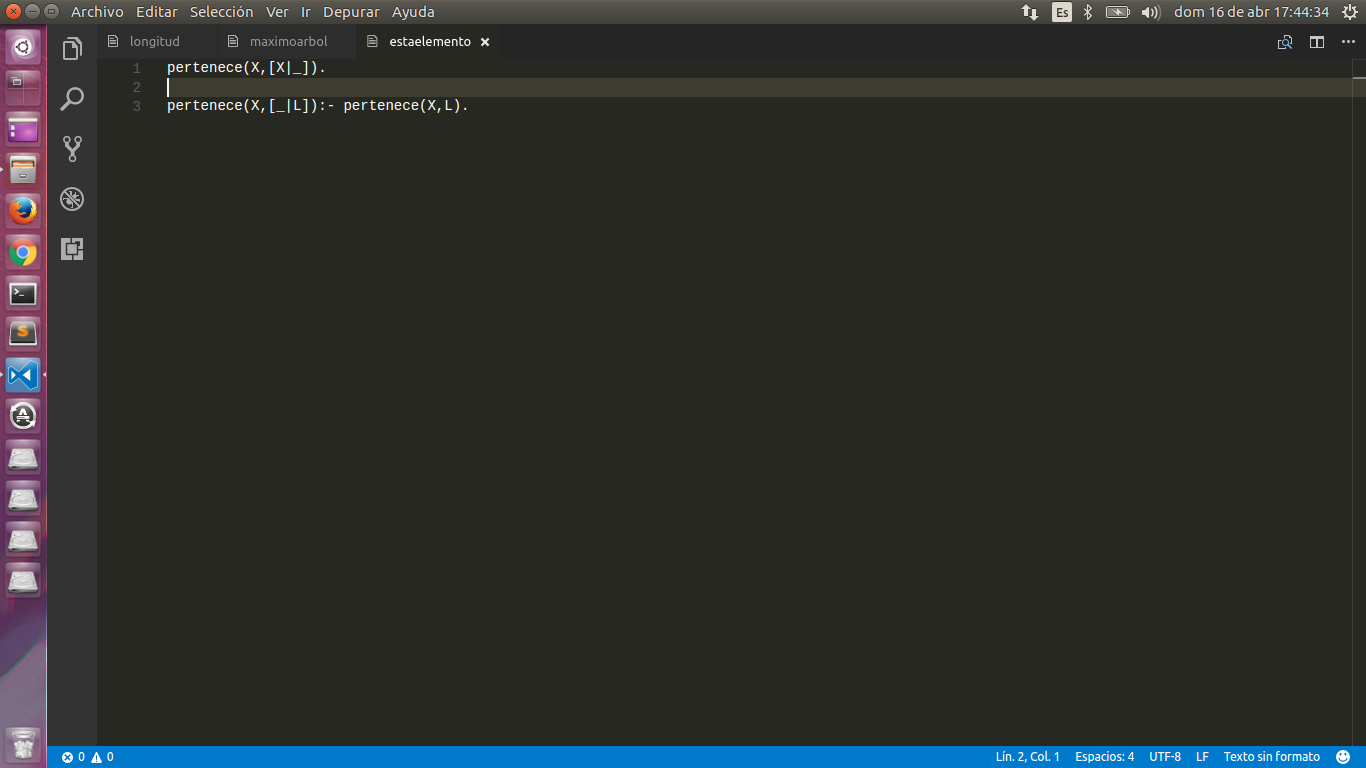


Código 4.- PROLOG

**Función Máximo Elemento de un Árbol.**

L1: Es un *hecho*, cuando la función encuentre un nulo, retornara un 0.

L3: Es una *regla,* recibe un árbol (t) con las variables I (izquierda), R (raíz), D (derecha), y se retornara el valor máximo en la variable M. Luego mediante la recursividad comienza a comparar estos tres nodos y guarda el mayor.



Código 5.- PROLOG

**Función Elemento Pertenece a la Lista.**

L1: Es un *hecho*, la función termina cuando el elemento X (que es el que se busca) es encontrado.

L2: Es una *regla,* empieza a recorrer la lista revisando el cabezal de esta y luego por recursividad llama a la cola de la lista, de modo que la recorre hasta que se encuentre el elemento, o que la lista quede vacía.

**CONCLUSIÓN**

Se presentaron dos lenguajes de programación que el alumno no conocía con anterioridad.

Comparando los dos lenguajes, LISP es más intuitivo y *user-friendly* que PROLOG, debido a su sintaxis y uso similar a los lenguajes de programación genéricos.

Las funciones pedidas, si bien eran básicas, cumplen su cometido ya que al ser así, es más sencillo y rápido comprender como funcionan estos dos lenguajes, en especial LISP debido a que el número de funciones a realizar es superior a las requeridas en PROLOG.

En lo personal prefiero trabajar con LISP, como se mencionó antes, su uso es más intuitivo y su sintaxis no es tan compleja de entender y una vez acostrumbrado a eso, se puede programar fácilmente en el (como fue el caso de la función que obtiene las soluciones de una ecuación cuadrática).