Tarea Programada #2

- La tarea debe entregarse al profesor a través del TEC Digital, el día y a la hora convenida.
- La tarea debe contener lo siguiente:
    a. Fuentes, todo el código necesario para ejecutar la tarea. El código debe estar debidamente documentado.
    b. Makefile para poder compilar los fuentes.
    c. Documentación, incluyendo al menos:
        i. Documentación en el código
        ii. Explicación del diseño y Arquitectura
- La tarea debe ser programada en YASM (Ensamblador) para Linux Ubuntu 12.X, en la sintaxis de Intel de x64.
- Toda tarea debe ser defendida ante el profesor, de tal manera todos los estudiantes deben poder explicar la solución satisfactoriamente.
- ¡Buena Suerte!

A Evaluar	**Puntos**	**Nota**
Documentación	10	
Validación de Errores y Casos especiales	10	
Resolución satisfactoria de las expresiones	45	
Impresión de la solución	25	
Completitud	10	
Total	**100**	
Función de Potencia	5	
Función de Módulo	5	
Total + Extra	**110**	

Evaluador de Expresiones Matemáticas

Evaluar funciones matemáticas siempre ha sido importante, se usan para toda clase de aplicación, desde los cálculos que hacemos para calcular nuestro presupuesto, hasta los cálculos para describir las partículas fundamentales en la mecánica cuántica, es por esto que se diseñará un software en ensamblador que nos ayude a resolver dichas funciones.

El software deberá leer una expresión matemática expresada con incógnitas; además de los valores de esas incógnitas desde el standard input. Por ejemplo, el estándar input tendría algo como:

2x + (6 – y) * 3x – ((x*z / 8) + 25),
x = 50,
y = 5,
z = 20

El primer parámetro hasta la primera coma siempre es la expresión qué puede tener desde 0 (en cuyo caso debe dar error) y hasta 1024 caracteres de largo, después vienen n cantidad de variables (de 0 hasta 20), correspondiente a las incógnitas de la expresión separadas por coma.
Note que:

a. El largo de la expresión es variable hasta 1024.
b. Hay n cantidad de variables, donde n es un número de 0 a 20. Ósea puede haber de cero hasta 20 variables diferentes.
c. El último elemento no tiene coma; ósea, que en caso de que no hayan incógnitas, no hay ninguna coma.

Como resultado se debe desplegar en pantalla el resultado de la expresión final. En el ejemplo anterior el resultado sería:

a. 2x + (6 – y) * 3x – ((x*z / 8) + 25)
b. (2x) + (6 – y) * (3x) – (((x*z) / 8) + 25)
c. (2*50) + (6 – 5) * (3*50) – (((50*20) / 8) + 25)
d. 100 + 1 * 150 – ((1000 / 8) + 25)
e. 100 + 1 * 150 – (125 + 25)
f. 101 * 150 – 150
g. 101 * 150 – 150
h. 15150 – 150
i. **15000**

Note que es necesario desplegar el procedimiento de resolución; el caso anterior es solo un ejemplo pero se debe diseñar alguna manera de imprimir en el standard output la resolución, esta debe impresa en Infix, sin importar cual se use para su resolución.

Consideraciones:

1. El programa debe validar la expresión, si esta no tiene sentido, como los siguientes ejemplos debe desplegar un mensaje de error.
    a. 2 ++ x
    b. 6x + 1 –
    c. (2x + 1
    d. (x + 1))
2. El resultado máximo deberá ser un número con signo de 64 bits, si el resultado es mayor a este, debe desplegar un mensaje de error.
3. Pueden haber de 0 a 20 incógnitas distintas en la expresión, de haber más debe enviar un error.
4. Las operaciones validas son:
    a. Suma (+)
    b. Resta (-)
    c. Multiplicación (*)
    d. División (/)
5. Se debe considerar en el caso de la división, que si el divisor es cero debe desplegarse un error. También se debe recordar que en el caso de la división ningún argumento debe ser cero en la instrucción.
6. En el caso de la multiplicación se debe considerar que puede estar tácita, como por ejemplo:
    a. $2x = 2 * x$
    b. En el caso de dos variables unidas, xy, la multiplicación se hace explicita x * y; sino se toma como una sola variable.
7. Los paréntesis son válidos y cualquier anidamiento de ellos también.
8. La expresión puede tener "Enters" cambios de línea o espacios, el final de la expresión lo marca la coma.
9. Todas las operaciones son enteras, no hay punto flotante.
10. Todas las variables deben existir en la entrada de lo contrario envía un error.

Puntos Extra

1. **Se dará valor extra si y solo si todos los otros puntos de la tarea funcionan bien.**
2. Implementar la operación de exponente con el operador "**", por ejemplo:
    a. $2 ** 5 = 2^5 = 32$
3. Implementar la operación de modulo con el operador "%", por ejemplo:
    a. $5 \,\%\, 2 = 1$