Algoritmos y Estructuras de Datos
Enunciado Primer Parcial Año 2018

Realizar un programa que evalúe una expresión aritmética con notación tradicional. La expresión contiene solamente números enteros, las cuatro operaciones aritméticas básicas y paréntesis. La expresión está compuesta por una serie de "*tokens*", donde cada *token* es un número, un operador aritmético o los paréntesis.

La evaluación de la expresión es la tradicional de izquiera a derecha, pero todas las operaciones tienen igual prioridad. Ejemplo:

$$5 - 2 * 2 = 3 * 2 = 6$$

Por lo que para definir prioridades en las operaciones debe utilizar paréntesis. El ejemplo anterior, bajo el orden de prioridad aritmético tradicional, debe escribirse como:

$$5 - (2 * 2) = 5 - 4 = 1$$

Para resolver la expresión debe leer la lista de tokens y utilizar la estructura de Pila, donde se almacenen tokens que contienen el primer operando y el simbolo de la operación. Ante la lectura del segundo operando, debe desapilar el operador y el primer operando, evaluar, y apilar el resultado. La existencia de varias operaciones implica la utilización como primer operando del resultado parcial obtenido, como es mostrado en el primer ejemplo.

Si el token tiene un paréntesis que abre, esto genera una nueva expresión, y deberá apilar en la pila principal un enlace a una nueva pila que contenga la nueva expresión, la cual será evaluada de igual forma, hasta que llegue un paréntesis que cierra, lo cual implica que la expresión ha terminado, y debe devolver el resultado de la subexpresión a la expresión llamadora original.

Un número negativo se representa entre parentesis con el signo menos antes de la cifra, por ejemplo:

$$5 - (2 * (-2)) = 5 - (-4) = 9$$

En este caso, al abrir el paréntesis del número negativo se genera la nueva pila, pero se apila en forma inicial solamente el signo menos. Al encontrar el parentesis que cierra al número -2 se devuelve el resultado del número negativo. Tenga en cuenta que este tipo de excepción es válido solo para el signo menos, no para los otros signos de las operaciones.

El resultado de la expresión es el tope de la pila luego de haberse leido todos los tokens de la lista.

Los clases que deberá utilizar para resolver el problema son:

- Token: que contiene un número, o un operador o un enlace
- Lista de Tokens: que contiene la expresión a evaluar
- Pila: que contiene tokens y es utilizada como auxiliar para el procesamiento

Deberá codificar la solución usando Clases de C++, con métodos que respeten el comportamiento expuesto.

La expresión puede ser dada desde teclado o bien leida desde archivo, independiente de la clase calculadora e invocada desde el método *main* del programa.

Es un *bonus* definir una función verificar que devuelva falso si la expresion no es correcta.

Podrá utilizar el código de las clases lista y pila dados en clases prácticas. El tipo de dato contenido en las clases Lista y Pila debe ser de la clase *Token*. Esta deberá tener un método que determine que tipo de valor contiene el token.