Universidad de Costa Rica
CI-1201 Programación II G5 - Profesor: Rubén Jiménez Goñi
Entrega: 10 de julio antes de medianoche por Mediación Virtual.

Proyecto Final (20%)

Resuelva el siguiente problema que se le plantea:

- Imagine que usted tiene un archivo con una serie de operaciones matemáticas y necesita saber los resultados. Los operadores pueden ser: sumas, restas, divisiones, multiplicaciones, elevaciones, logaritmos (ln), sen, cos, tan y una función de 3 parámetros escogida por usted. Asimismo, algunas operaciones tienen operaciones agrupadas dentro de paréntesis, dentro de los cuales puede haber más agrupaciones hasta cualquier profundidad. Cree una aplicación capaz de abrir dicho archivo, procesar línea por línea cada operación y escribir los resultados en otro archivo. Al procesar el archivo, su aplicación debe respetar las reglas de precedencia de operadores y paréntesis.

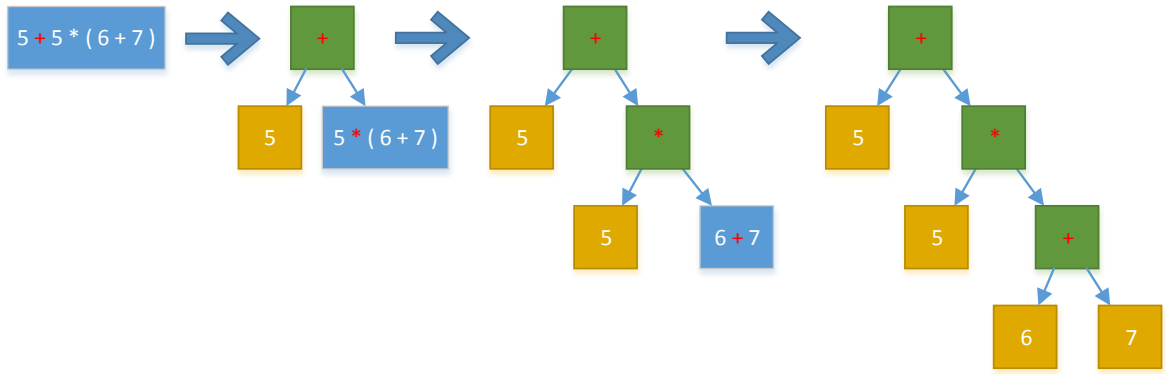
La solución de su examen está restringida a las siguientes reglas:

1. No puede utilizar las clases de la *stl* (*Standard Template Library*). Cree sus propias estructuras de datos basados en el *Árbol* y la *Lista* emplantillada que hemos visto en clases.
2. Sus estructuras de datos (árbol y lista) deben estar emplantilladas. Utilice la idea del Nodo que se ha manejado en los ejemplos.
3. Cuando procesa el archivo de entrada, debe crear una *Lista* de operaciones.
4. Cada vez que procesa una operación, agrega el resultado de la misma en una *Lista* y luego la escribe en un archivo.
5. La jerarquía de clases es la siguiente:
    a. *Elemento*: del cual heredan:
        i. *Operacion*: Recibe un string en el constructor.
        ii. *Operando*: Es un número double
        iii. *Operador*: Para representar los operadores. Tiene un método operar que recibe una lista de elementos (en vez de exclusivamente 2 parámetros). Del cual heredan:
            1. *OperadorUnario*: Por ejemplo las operaciones de logaritmo y las trigonométricas
            2. *OperadorBinario*: Por ejemplo las operaciones de sumas, restas, divisiones, multiplicaciones y elevaciones.
            3. *OperadorTernario*: La función de 3 parámetros definida por usted. Para facilidad de procesamiento debe verse algo así NOMBRE_FUNCION[OPERACION_1, OPERACION_2, OPERACION_3]
6. Para resolver cada operación no puede utilizar una *Pila*. Debe utilizar un A*rbolDeOperadores.* Su árbol tiene las siguientes peculiaridades:
    a. No es unario, ni binario ni ternario.
    b. Debe tener un nodo, pero el nodo no tiene punteros ni a hijo izquierdo, ni derecho. Debe tener una lista de hijos (el tipo de dato es su propia lista emplantillada).
    c. Debe ser capaz de imprimir el árbol a partir de cualquier nodo del mismo.

7. Procese el árbol de la siguiente forma:
    a. Inicialmente su árbol contiene una operación completa en un objeto *Operacion*.
    b. Posteriormente cree un método que procese su árbol de la siguiente forma:
        i. Si un nodo es una operación, busque el operador de menor precedencia y divida la operación en: uno, dos o tres operaciones (u operandos) y un operador según corresponda. Ponga el operador en el lugar de la operación y cree una lista de hijos y asígnelas al operador.
        ii. Si el nodo es un operando u operador, no haga nada
        iii. Continúe haciendo el mismo proceso mientras haya operaciones dentro del árbol.
        iv. Considere que los paréntesis tienen la precedencia más alta.
8. Cada vez que procese un nodo del árbol a la hora de solucionarlo, imprima el árbol completo. El objetivo de esto es ver cómo se van resolviendo las operaciones.
9. Puede resolver el proyecto individual o en parejas pero debe venir claramente específico en la entrega de su proyecto. Si lo hace en pareja, solo uno de los dos entrega el proyecto en Mediación Virtual (pero asegúrese de que su pareja lo entregue).
10. PUNTOS EXTRA (10%): Implemente un iterador para su *ArbolDeOperadores*.

Ejemplo:

- Creación del árbol:



- Solución del árbol