# Parentheses Match

The task is to determine whether the grouping symbols--parentheses, brackets, curly braces, etc.--in an arithmetic expression, such as ***"***[ ( 5 + 7 ) \* 3 ] )***"***, match each other. (How would you do this without a stack?)

## Algorithm

The arithmetic expression can be treated as a string of digits, operators, spaces, and grouping symbols. Process the string one-by-one. Ignore everything except the grouping symbols and *push* the left symbols on the stack. When you get to a right symbol, *pop* the stack (what if the stack is empty?). Somehow determine whether the left symbol “matches” (not “equals”!) with the right symbol. List all the cases that return false:

If the expression has passed all those tests, the grouping symbols match each other. Return true

## Assignment

Make an ArrayList of expressions with grouping symbols and output whether the grouping symbols match. Test for ( ), [ ], { }, and < >. Type in the test data as shown in the sample run below. Use only the stack methods in java.util.Stack<E>. Since the Java Collections classes always store objects, not primitives, you could process either a stack of Strings or of Characters. The shell below uses Strings.

Nice way to specify the grouping symbols.

public class ParenMatch  
{  
 public static final String LEFT = "([{<";  
 public static final String RIGHT = ")]}>";  
   
 public static void main(String[] args)  
 {  
 System.out.println("Parentheses Match");  
 ArrayList<String> parenExp = new ArrayList<String>();  
 /\* enter test cases here \*/  
   
 for( String s : parenExp )  
 {  
 boolean good = checkParen(s);  
 if(good)  
 System.out.println(s + "\t good!");  
 else  
 System.out.println(s + "\t BAD");  
 }  
 }  
 //returns the index of the left parentheses or -1 if is not  
 public static int isLeftParen(String p)  
 {  
 return LEFT.indexOf(p);  
 }  
 //returns the index of the right parentheses or -1 if is not  
 public static int isRightParen(String p)  
 {  
 return RIGHT.indexOf(p);  
 }  
 public static boolean checkParen(String exp)  
 {  
   
 }  
}

***Sample Run***

Parentheses Match  
5 + 7 good!  
( 15 + -7 ) good!

) 5 + 7 ( BAD  
( ( 5.0 - 7.3 ) \* 3.5 ) good!  
< { 5 + 7 } \* 3 > good!  
[ ( 5 + 7 ) \* ] 3 good!  
( 5 + 7 ) \* 3 good!  
5 + ( 7 \* 3 ) good!  
( ( 5 + 7 ) \* 3 BAD  
[ ( 5 + 7 ] \* 3 ) BAD  
[ ( 5 + 7 ) \* 3 ] ) BAD  
( [ ( 5 + 7 ) \* 3 ] BAD  
( ( ( ) $ ) ) good!  
( ) [ ] good!

Missing a number! but this algorithm only checks grouping symbols.