Middle East Technical University
Department of Computer Engineering

CENG 111

Fall 2014

Take Home Exam 3
[**Version: 1.2**]

REGULATIONS

Due date: 25 December 2014, 23:59, Thursday *(Not subject to postpone)*

Submission: Electronically. You should put your solution code into a file called `the3.py` and submit it through the COW web system. Resubmission is allowed (till the last moment of the due date); the last one replaces the previous ones.

Team: There is **no** teaming up. The take home exam has to be done/turned in individually.

Cheating: **This is an exam:** All involved parties (source(s) and receiver(s)) get zero, and will be subject to disciplinary action.

INTRODUCTION

After a successful contact with the planet Zoitank in the year 2030, due to Zoitankians being very friendly and peaceful with Earthians, tremendous commerce between the two planets immediately followed. However, very soon, a big problem emerged: Zoitankians had a different interpretation of expressions involving operators.

PROBLEM

In this THE, your task is to solve the expression conversion problem such that commerce between the two planets can commence. You should implement two functions described below:

- `ToZoitankian(EarthianExpression)`, which produces the Zoitankian version of the Earthian expression such that, when evaluated by a Zoitankian, the Zoitankian expression produces the same result as `EarthianExpression` when evaluated by an Earthian.
- `ToEarthian(ZoitankianExpression)`, which produces the Earthian version of the Zoitankian expression such that, when evaluated by an Earthian, the Earthian expression produces the same result as `ZoitankianExpression` when evaluated by a Zoitankian.

The list of operators used by Zoitankians and Earthians are given in Table 1 and 2:

Table 1: *Precedence and associativity of Zoitankian operators. Top: highest precedence, bottom: lowest precedence.*

Operator	Type	Associativity	Description
`+`	Binary	Left-to-right	Addition
`/`	Binary	Left-to-right	Multiplication
`-`	Binary	Left-to-right	Subtraction
`*`	Binary	Left-to-right	Division
`^`	Binary	Right-to-left	Exponentiation

Table 2: *Precedence and associativity of Earthian operators. Top: highest precedence, bottom: lowest precedence.*

Operator	Type	Associativity	Description
^	Binary	Right-to-left	Exponentiation
*, /	Binary	Left-to-right	Multiplication, Division
+, -	Binary	Left-to-right	Addition, Subtraction

Once you carefully analyze the two tables, you should realize that the operators have different meanings and precedences although they have the same associativities. Don't be scared though since, in Zoitankian and Earthian expressions, parentheses have the same functionality in expression evaluation, which you will exploit in solving the conversion problem.

SPECIFICATIONS

- The arguments to the functions `ToZoitankian()` and `ToEarthian()` are strings composed of the operators given in the preceding tables and as operands, single lowercase letters from the English alphabet (i.e., a, b, c, ..., z).
- The input (and of course the output) expressions might contain parentheses, and in Zoitankian and Earthian expressions, parentheses have the same functionality.
- The operands and the operators may or may not be separated by a space character. However, it is guaranteed that tab, newline or other whitespaces will not be used.
- The outputs of the functions should be strings (as shown in "Example Run") and not contain any whitespaces.
- Your functions should just return values and not print anything on screen.
- You can use recursion and iteration as freely as possible.
- You are not allowed to import any libraries or functions.

EXAMPLE RUN

```
>>> ToZoitankian('a *( x ^ y ) ^ z / ( b - c )')
'a/((x^y)^z)*b-c'
>>> ToEarthian('a *( x ^ y ) ^ z / ( b - c )')
'(a/x^y)^(z*(b-c))'
>>> ToZoitankian("a / ( b + c ) * y ^ x")
'(a*b+c)/(y^x)'
>>> ToEarthian('a / ( b + c ) * y ^ x ')
'(a*(b+c)/y)^x'
```

Note that the output expressions were removed of some redundant parentheses; however, we will not expect your code to do that.

EVALUATION

- There will be no erroneous test input.
- All submissions will be tested under strictly equal conditions on multiple data sets on Linux-running lab machines using Python version 2.7.6.
- It is acceptable if your output expressions contain redundant parentheses. While grading your solutions, our automated evaluation system will just check whether the evaluation order is correctly reflected.