COSC2430: Programming and Data Structures
Expression Evaluation Using Stacks

1 Introduction

You will create a C++ program to evaluate expressions combining set union, set intersection and parentheses. The program must exploit a stack to convert the matrix expressions from infix notation to postfix notation and a second stack to evaluate the postfix expression. In order to maintain similar notation to arithmetic expressions, + will denote union and * will be intersection. You will evaluate expressions like$A * B + C$, $(A + B) * (D + E * F)$ or $A * B + (C * D)$, where A, B, C and D are sets.

We will define the operation "+" of two sets A and B as the union of the contents of both sets:

$$A + B \equiv A \cup B \tag{1}$$

Similarly, we will define the operation "*" of A and B as the intersection of A and B:

$$A * B \equiv A \cap B \tag{2}$$

These definitions allow us to write long algebraic equations. We assign to the "+" a lower precedence than "*" (in the same way as we do in algebra) and parentheses change the precedence of an expression.

2 Input and Output

The input is a regular text file, where each line will contain an arithmetic expression and terminate with an ' n' character.

Input example:

```
expression1.txt
-----------------------------------------------
(A+B) + C
(+A+(B))*C
(A*(B*C))*C
```

The set A, B and C are specified in the file "A.txt", "B.txt" and "C.txt", respectively. Each set contains a list of integers (¡10000).

```
A.txt
-----------------------------------------------
-6 1 2 3 4 5 6
<EOF>

B.txt
```

```
-----------------------------------------------
1 2 7 8<EOF>

C.txt
-----------------------------------------------
1 9 7
<EOF>
```

The output should be sorted.

Output example:

```
(A+B) + C=-6 1 2 3 4 5 6 7 8 9
(+A+(B))*C=1 7
(A*(B*C))*C=1
```

3 Program input and output specification

The main program should be called expression. The output should be written to the console (e.g. printf or cout), but the TAs will redirect it to create some output file.

Call syntax at the OS prompt (notice double quotes):

```
expression input=<file name>"
```

Example of program call:

```
expression input=expression1.txt
```

Assumptions:

- Keep in mind a single + can be a sign instead of an operator.

4 Requirements

- std::stack is not allowed to be used in the homework. Please implement your own stack.
- Correctness is the most important requirement: TEST your program with many input files. Your program should not crash or produce exceptions. We only accept program that can run on the server, please test your program on the server.
- Stacks are required to convert the expression from infix to postfix and to evaluate the postfix expression.
- Your program should get the result within 2 seconds.