CSE 4250: Project #3, Expression Evaluation

Due: Friday, 1 November 2019

The Task

Evaluate expressions in reverse Polish notation using a generic stack. It is required that you use Ada's doubly-linked list for your stack, but simplify the interface to that of a generic stack as described in the Ada 2005 rationale.

Use the stack to evaluate expressions in star semirings. In this project there are two different kinds of expressions--two different star semirings. The syntax of the both kinds of expressions is the same, but the evaluation is different.

```
domain    0        1       +     .       *
Boolean False    True     or   and    _->True
Natural  inf       0      min   +     _->inf
```

Tropical Star-Semiring

The evaluation is somewhat unusual using the (min,+) tropical semiring. There are two constants denoted by the character 0 and the character 1. The character 0 denotes infinity — the largest natural number. The character 1 denotes zero — the smallest natural number. The single digits 2 through 9 represent the result of applying the successor function (the plus 1 function) 2 through 9 times, respectively. (For simplicity in this project, we have no representation for the successor of zero.) There are three operations. Two are familiar, binary operations: min (denoted '+') and addition (denoted '.'). The third operation (denoted '*') — a unary operation — always returns infinity.

Boolean Star-Semiring

In the Boolean star-semiring there are two constants denoted by the character 0 and the character 1. The character 0 denotes false, and character 1 denotes true. The single digits 2 through 9 represent the result of applying the successor function (the plus 1 function) 2 through 9 times, respectively. (For simplicity in this project, we have no representation for the successor of zero.) There are three operations. Two are familiar, binary operations: disjunction (denoted '+') and conjunction (denoted '.'). The third operation (denoted '*') — a unary operation — always returns true.

Input/Output

Read from the standard input stream and write to the standard output stream. On each line of the input there will be one correctly formed expression tagged with either Z for the tropical star-semiring or B for the boolean star-semiring. For simplicity all the operators and operands are

exactly one character. In the input, the characters may or may not be separated by white-space; There is not such thing as an empty expression — an expression must have at least one character in it. The output is the result of evaluating the expression — a single value one for each line.

Here are several sample expressions below on the left. To the right is the corresponding result. These do not appear in the input! The comments in brackets do not appear in the input or the output!

```
Z  7 8 .          15    [addition]
Z  7 8 +          7     [min]
Z  2 3 4 . .      9     [2+(3+4)]
Z  2 3 . 4 .      9     [(2+3)+4]
Z  0              0     [infinity]
Z  1              1     [zero]
Z  0 1 +          1     [zero]
Z  1 1 +          1     [zero]
Z  1 1 .          1     [zero]
Z  1 0 * .        0     [infinity]
```

Spaces are immaterial and should be ignored. You may assume the length of the longest line is 100 characters.

```
Z  78.            15    [addition]
Z  78  +          7     [min]
Z  234..          9     [2+(3+4)]
Z  2     3.4.     9     [(2+3)+4]
Z           0     0     [infinity]
Z    1            1     [zero]
Z  01 +           1     [zero]
Z    1  1   +     1     [zero]
Z  11 .           1     [zero]
Z    10     *.    0     [infinity]
```

Read until the end of the input stream—one epxression per line. *The input is a mixture of the two kinds of expression.*

```
B  7 8 .          1     [conjunction]
B  7 8 +          1     [disjunction]
B  2 3 4 . .      1     [2 and (3 and 4)]
B  2 3 . 4 .      1     [(2 and 3) and 4]
B  0              0     [false]
B  1              1     [true]
B  0 1 +          1     [true]
B  1 1 +          1     [true]
B  1 1 .          1     [true]
B  1 0 * .        1     [conjunction of trues]
B  78.            1     [conjunction]
B  78  +          1     [disjunction]
B  234..          1     [2 and (3 and 4)]
B  2     3.4.     1     [(2 and 3) and4]
B           0     0     [false]
B    1            1     [true]
B  01 +           1     [true]
```

```
B    1  1   +    1     [true]
B   11 .         1     [true]
B    10     *.   1     [conjunction of trues]
```

GNAT

Your program will be compiled roughly like this:
`gnatmake eval`
Make sure you adhere to the Ada 2005 or later standard. Details about running `gnatmake` can be found on the Internet.

Also the GNAT compiler will check your style to a large degree.

```
gnatmake -gnaty3abcehiklmprM90 eval
```
Do not use tabs. One normally recommends indenting three or four spaces, but since we cannot check "three *or* four", please use three.

There are several additional options that we consider obligatory:

```
gnatmake -gnaty3abcehiklmprM90 -gnata -gnatwe -gnato -gnatVa  eval
```

Turning it in

You may work in groups of no more than two people. You may not work with the same person on any of the remaining projects. If in a group of two, make sure both names are in the program and turn in the assignment from just one account. By the due date, turn in your source program. At the beginning of your source file include the following header information in comments:

```
-- Author: name1, e-mail address
-- Author: name2, e-mail address
-- Course: CSE 4250, Fall 2019
-- Project: Project #3, Expression Evaluation

-- This program compiles with gnat version :
-- GNAT 7.3.0
-- GNAT GPL 2017 (20170515-63)
```

Be sure to include the version of the GNAT compiler that compiles your code. You can get the version by running:
`gnat --version`