FINAL PROJECT

COMPILERS

May 2022

1. The Taxy way

Create the syntax analyzer of a small language to recognize movement statements into a Cartesian plane. Your program should read from standard input.

The input consists of a single line containing a path that is a sequence of four possible instructions:

- UP n
- DOWN n
- RIGHT n
- LEFT n

UP, DOWN, RIGHT, LEFT are the directions along the axis, n a positive integer that represents how many unitary steps must be done in the specified direction.

As output, the parser checks whether the inserted path is or not a valid path for moving from one assigned point, e.g. A(0,0), to one other assigned point, e.g. B(5,6).
To reach B from A there are infinite valid paths, example of valid paths are:

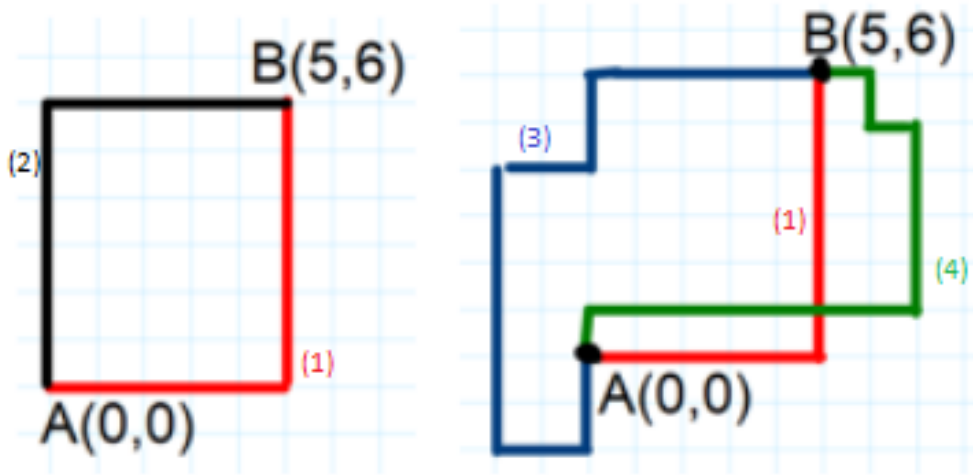
(1) RIGHT 5 UP 6

(2) UP 6 RIGHT 5

(3) DOWN 2 LEFT 2 UP 6 RIGHT 2 UP 2 RIGHT 5

(4) UP 1 RIGHT 7 UP 4 LEFT 1 UP 1 LEFT 1

The 4 possible input above corresponds to the 4 valid paths in figure:



When the inserted path is wrong you can either print an error message, or even print a possible couple of moves to reach the destination B.

2. Translate the infix calculator notation into an abstract syntax tree (AST)

Create the syntax analyzer of a calculator with infix notation.

The calculator grammar is as follows:

$$
\begin{aligned}
E &\rightarrow E + T \mid E - T \\
&\mid T \\
T &\rightarrow T * F \mid T / F \mid T \,\%\, F \\
&\mid F \\
F &\rightarrow \textbf{num} \\
&\mid -F \mid +F \\
&\mid (E)
\end{aligned}
$$

The compiler will need to translate the input into infix calculator notation to an AST.

To construct an AST, you have to handle multiple types, say integers and pointers to tree nodes. For the following expression, for example:

```
+ 25 * - (2 - 3)
```

your program should produce the following output:

```
*
  (+)
    25
  (-)
    -
      2
      3
```

Note that the unary operators are printed enclosed by a pair of parentheses.

Your program should read from standard input. The input consists of a single line containing an arithmetic expression. The program should print to standard output the tree with space indentations (all the nodes at the same level have the same number of space indentations), as in the following table:

Input	Output
+␣25␣*␣-␣(2␣-␣3)	* ␣␣(+) ␣␣␣␣25 ␣␣(-) ␣␣␣␣- ␣␣␣␣␣␣2 ␣␣␣␣␣␣3
(-␣11␣+␣7)␣*␣(31␣-␣+␣12)	* ␣␣+ ␣␣␣␣(-) ␣␣␣␣␣␣11 ␣␣␣␣7 ␣␣- ␣␣␣␣31 ␣␣␣␣(+) ␣␣␣␣␣␣12

Alternatively the tree can be printed in json or xml output.