XMini Programming Language Interpreter

For our first project we will write an interpreter for our own little toy programming language. The language will have loops, conditional statements. We're looking for just straight-line code consisting of assignment statements and output statements (i.e., `print`). Our language is designed to be very easy to parse. It bears no resemblance to any real programming language. Details of the language are shown below.

You will be given relatively little starter code for this project. A large part of the assignment is for you to design your own approach, including data structures, algorithms, structs/classes, functions, etc. Part of your grade will be based on good choices of data structures and algorithms as well. You can use any code developed in class, and you can naturally write any code you want yourself.

1 XMini Language

We ask that you implement four language statements. Two of these (`var` and `set`) are for creating and assignment values to variables, and two of these (`output` and `text`) are for producing output (i.e., printing to the screen). A XMini program consists of zero or more statements. Each statement begins with a keyword (`var`, `set`, `output`, or `text`).

- `text` statements result in a text message being displayed on the screen. The text message can be either a single word, or a quoted string. The following are examples of legal text statements:

  text Hello

  text "Hello World"

- `output` statements result in a number being displayed on the screen. The number displayed is the result of evaluating any legal XMini expression. For now, XMini only supports integer variables, and so only integer values can be displayed. The following are examples of output statements. For more information on XMini expressions, see below

  output 42

  output + 1 1

  output * x x

  output + * x x * y y

- `var` statements initialize a new variable. If a var statement specifies a variable that already exists, then XMini must generate a warning (not an error), with the text, "variable <varName> incorrectly re-initialized", where <VarName> is the variable's name. Regardless of whether the warning message is created, the result of a var statement is to ensure that there is a variable in the XMini internal memory with the specified value. The variable must be set to the value of a legal XMini expression. The syntax is: "var <varName> <expr>". The following are examples of legal var expressions:

  var x 42

  var y + x 1

- `set` statements are just like `var` statements, except a `set` statement should be used to re-assign a new value to an existing variable. As such, if a `set` statement specifies a variable that does not already exist, then XMini must generate a warning (not an error) with the text, "variable <varName> not declared". Regardless of whether the warning message is created, the result of a `set` statement is to ensure that there is a variable in the XMini internal memory with the specified value. The following are examples of legal `set` expressions:

  set x 42

  set x + x 1

XMini programs do not have to have only a single statement on a line, in fact, line breaks don't really matter to XMini (nor do tabs or spaces). Additionally, XMini programs can have comments between statements (but not in the middle of a statement). For example, the following is correct XMini:

```
text "Hello, and welcome to this world" // first line
var x 0
// set x to (5 + 3) * (x - 1)
set x
    *
        + 5 3
        - x 1
```

A comment is marked by // and continues to the end of the current line. Note that the following statement is illegal in XMini, since we can't have comments in the middle of a statement:

```
output + // two terms in this expression
       * 5 3 // 5 * 3 is the first term
       - 10 7 // 10 - 7 is the second term
```

1.1 Variables and the XMini Memory

To implement variables, you will need to have a symbol table. A symbol table is like a little database that keeps track of the values of all the variables in your program. You should choose a reasonable data structure for your symbol table - one that will scale to potentially very large number of variables. It is not necessary to explicitly model memory (i.e., addresses and memory locations and things like that). However, you will need to implement some way of knowing that variable x was set to 42. How you implement this is up to you.

1.2 XMini Expressions

XMini has a fairly rich set of C-like integer operators, but has a distinctly non-C expression syntax. The reason for the odd syntax is so that XMini is easy to parse. Perhaps the easiest-to-parse languages are those that use prefix expression syntax, so that's what XMini uses. For example, instead of writing 2 + 2 (infix expressions), we write "+ 2 2" in XMini. The order for the arguments is important for non-commutative operations, so, for example, the XMini expression / 10 5 will evaluate to 2 (10 / 5 is 2). The complete list of operators that you must support are:

- Binary math operators: +, -, *, /, %
- Binary logic operators: &&, ||
- Comparison operators: <, >, ==, !=, <=, >=
- Unary operators: !, ~

Note that for logic operations, we use the C conventions for integer values. Any value other than zero is `true`, and zero is `false`. When evaluating an expression that produces a logical value (e.g., && or <=), you must evaluate `true` expressions to 1 and `false` expressions to zero. So... && 5 42 evaluates to 1 and + && 6 12 10 evaluates to 11

Note that we are not implementing the bit-wise logic at this time. Please implement $\sim$ as arithmetic negation (i.e., $\sim$ 5 is really -5).

NOTE: in XMini there must be at least one space (or tab or newline) between each operator and operand. For example " 5" is an error. "$\sim$ 5" is the correct way to write -5 in XMini.

1.3 Running XMini

To actually run XMini, please write your own `main()` function that will read and execute the XMini program.

Input to your main is the location of the XMini program to execute.

1.4 Important Submission Formatting

- You should no use third-party library (only standard Java libraries).
- You should use Java 17.
- Inlcude a bash script (called s) that will compile and run your program with a single argument (path to an input XMini program to execute).