Project 6: Implementing Scopes and Branching in SpartyTalk

DUE: November 28, 2022 11:59 PM EDT**. *Please make sure to always have backup copies of your files!***

Description

In this project, we continue implementing the language called *SpartyTalk* by adding boolean expressions, scopes and IF/IF-ELSE statements to our language. The implementation of this project should be based on the lexer implemented in Project 1, parser implemented in Project 3, and interpreter implemented in Project 5.

First, let us take a look at the improved *SpartyTalk*, which now has branching. The following is a sample program written in *SpartyTalk*:

```
gogreen;
    nvar a = -10.5;
    svar b = "hello\n";
    svar c = "world";
    svar d = b + c;
    nvar e = a * 2;
    nvar f = 3.5 / a;
    f = f / 7.5 * 3;
    spartysays "hi " + e;
    if a < f gogreen;
        spartysays "a is less than f";
    gowhite; else gogreen;
        spartysays "a is greater or equal than f";
    gowhite;
gowhite;
```

The language separates instructions with semicolons, just like in C/C++ or Rust. Instead of having a `main()` function, the starting point of execution in *SpartyTalk* is the `gogreen` instruction, and the end of the execution is determined by the `gowhite` instruction. The statements between `gogreen` and `gowhite` instructions constitute a *scope*.

SpartyTalk is a *strongly-typed* language, and it does not do *type inference* like Python or Rust (i.e., guessing types based on the assigned value). Instead, it requires explicit specification of type during the declaration/initialization of a variable, just like in C/C++. We have two basic data types: *numbers* and *strings*. We use the `svar` keyword to declare a string variable and `nvar` keyword for declaring a numeric variable.

SpartyTalk does not like ambiguity. The variables must be initialized (e.g., assigned a value during the declaration). Also, our language is quite strict about type conversion. To convert a value from number to string, we use the **svar s = n;** syntax, where **n** is a numeric variable (not literal or expression). To convert a value from string to number, use the **nvar n = s;** syntax, where **s** is a string variable (not expression).

SpartyTalk produces output using the **spartysays** command. The argument of this command must be a string expression, a single number, single numeric variable, or an expression that evaluates to a string.

Also, our language currently supports four operations: **+**, **-**, *****, and **/**. If **+** is used with numbers, it performs arithmetic addition. If **+** is used with strings, it performs concatenation. If **+** is used between a string and a number, it converts the number to a string and performs concatenation (like in Python). Please check the tests carefully to understand the expected behavior of SpartyTalk expressions.

SpartyTalk now supports branching using **if** and **if...else** statements. An **if** keyword is immediately followed by a boolean expression, and this boolean expression is followed by a scope. Similarly, in the **if...else** statement, **if** is followed by a boolean expression, after which goes a scope (the true scope) followed by an **else** statement, after which goes another scope (the false scope). Please see the above example and the assignment tests to make sense of how the branching is expected to work.

In this project, you are expected to modify the grammar of the language to incorporate scopes, boolean expressions, and **if/if-else** statements to it. In this project, you will do the modification of grammar by yourself. You are expected to continue implementing the function **interpret_spartytalk().** The function has the same input-output specifications as in Project 5. Please note that **interpret_spartytalk()** is not a scope interpretation function. You are supposed to implement a scope interpretation function and invoke it inside **interpret_spartytalk()**.

For example, if the argument of **interpret_spartytalk()** is this program:

```
gogreen;
    nvar a = 10;
    nvar b = a * 2.2;
    if a < b gogreen;
        b = a;
    gowhite;
    spartysays b;
gowhite;
```

Then the result of the execution should be the following string printed to the standard output (followed by a newline, like in Python's **print()**):

```
10
```

Please examine the assignment's tests to learn how scope binding and shadowing works in *SpartyTalk*.

Implementation

Continue the implementation of **interpret_spartytalk()** function in **solution.py**. Implement the comparison operations **<**, **>**, **<=**, **>=**, **!=**, and **==**, as well as logical operations **and**, **or**, and **not**. Implement scopes, as well as **if** and **if...else** statements using these scopes.

Testing and Grading

The solution will be graded using 40 autograding tests: 20 tests in **`test_open.py`** and 20 additional hidden tests that will be used by the instructors while grading. The hidden tests *will not* introduce any new challenges on top of the ones already tested by the open tests. To run the tests, invoke the **`pytest`** command while in the project directory. Each test is worth 2 points, resulting in 80 total possible points. Please read the open tests to better understand the requirements of the implementation, *but do not modify the tests*. **Submit the solution to D2L.**

Have fun!