Sinirli Turkce Programlama Dili Derleyicisi - SiNiR

CmpE 150.03 Introduction to Computing, Fall 2020
Homework 5 – Due: 28/12/2020, 23.59

In this homework, you are going to implement the SiNiR compiler. The details are as follows:

Given a file named **calc.in**, your program should check whether there are any syntax errors.

- If there is a syntax error,
  - Create a new file, named **calc.out** and print “Dont Let Me Down” inside.
- If there are no syntax errors,
  - Create a new file, named **calc.out** and print “Here Comes the Sun” inside.

calc.in is composed of three parts. If one of those parts are missing, there is a syntax error.

```
AnaDegiskenler
<init-statement>
<init-statement>
<init-statement>
….
YeniDegiskenler
<mid-statement>
<mid-statement>
<mid-statement>
…..
Sonuc
<final-statement>
```

The general structure of **calc.in** should be in the following form:

- Empty lines are allowed anywhere in the file.
- The file should start with the AnaDegiskenler statement. If not, there is a syntax error.
- There are zero or more <init-statement>s after AnaDegiskenler statement. Each <init-statement> should be in a single line. If not, there is a syntax error.
- The file then continues with the YeniDegiskenler statement. If not, there is a syntax error.
- There are zero or more <mid-statement>s after the YeniDegiskenler statement. Each <mid-statement> should be in a single line. If not, there is a syntax error.
- The file continues with the Sonuc statement. If not, there is a syntax error.
- There is zero or one <final-statement> after the Sonuc statement. <final-statement> should be in a single line. If not, there is a syntax error.

Important:

- The language is case sensitive.
- Keywords cannot be used as variable names.

<init-statement>

- is constructed by:
  <var-name> degeri <value> olsun

- Any number of white-spaces (including space and tab) are allowed in <init-statement>.
- <var-name> is composed of at most ten alpha-numeric characters.
- The same variable cannot be assigned to more than once.
- <value> should be one of the following:
  - <digit>
  - {dogru, yanlis}
  - <digit>.<digit>
- <digit> should be one of the following:
  - {sifir, bir, iki, uc, dort, bes, alti, yedi, sekiz, dokuz}
  - {0,1,2,3,4,5,6,7,8,9}
- If above rules do not hold, there is a syntax error.

<mid-statement>

- is constructed by:
  <var-name> degeri <expression> olsun

- Any number of white-spaces (including space and tab) are allowed in <mid-statement>.
- <var-name> is composed of at most ten alpha-numeric characters.
- <expression> should be either <arithmetic-expression> or <logical-expression>.
- <arithmetic-expression> is a combination of <operand>s, <operator>s, and <parenthesis> that are separated from each other with white-spaces.
  - <operand> should be one of the following:
    - <arithmetic-expression>
    - <digit>
    - <digit>.<digit>
    - <var-name>
  - <var-name> should represent either an integer or a floating point number variable.
  - <operator> should be one of the following:
    - + - *
    - arti eksi carpi
  - <parenthesis> should be one of the following:
    - ( )
    - ac-parantez kapa-parantez
- <arithmetic-expression> cannot include logical operands or operators.
- <logical-expression> is a combination of <operand>s, <operator>s, and <parenthesis> that are separated from each other with white-spaces.
  - <operand> should be one of the following:
    - <logical-expression>
    - dogru yanlis
    - <var-name>
  - <operator> should be one of the following:
    - ve veya
  - <parenthesis> should be one of the following:
    - ( )

        - ac-parantez kapa-parantez
    - <var-name> should represent a boolean variable.
- If above rules do not hold, there is a syntax error.

<final-statement>

- is constructed by:
    <expression>

- The same rules for <expression> apply (as above).
- If above rules do not hold, there is a syntax error.

Submission: You will submit a single python file over Moodle. Your .py file should be named with the underscore character (_) followed by your student number (e.g. _2019700030.py).

Your homework will be evaluated in different categories so that you can collect **partial points**:

A	Correctly parse the file without any <init-statement>, <mid-statement>, <final-statement>	+5	If the code fails in this test case, the absolute grade will be **zero**
B	Correctly parse <init-statement>s composed of only integers	+10	
C	Correctly parse <init-statement> composed of combination of integers, floats and booleans	+15	
D	Correctly parse <mid-statement>s composed of only integers	+10	No variable is used on the right side of the statement (after degeri)
E	Correctly parse <mid-statement>s composed of combination of integers, floats, booleans	+15	No variable is used on the right side of the statement (after degeri)
F	Correctly parse <mid-statement>s composed of only integers and integer variables	+15	This requires also clearing B
G	Correctly parse <mid-statement>s composed of any operand	+15	This requires also clearing C
X	Correctly parse any file	+15	This requires also clearing G

Late Submission: Allowed with penalty: -% 10*(number_of_days_late)^2. Example case: You are 2 days late, and you got 90 from evaluation. You will get 90 * (1 - 0.4) = 54 as your final grade. You will get 0 if you are more than 3 days late.

Example calc.in files (more will be provided)

```
AnaDegiskenler
x degeri 4 olsun
x1 degeri 2 olsun
a degeri    dort olsun
b degeri uc nokta dort olsun
y degeri 8 olsun
```

z degeri 3.2 olsun
b1 degeri dogru olsun
b2 degeri yanlis olsun

YeniDegiskenler
t1 degeri x + (y) - z olsun
t4 degeri x arti ac-parantez y kapa-parantez eksi z olsun
t2 degeri (t4 + ac-parantez ((y)) - z) kapa-parantez olsun
t3 degeri x ve b1 ve (b2 veya b1) olsun

Sonuc
x arti ac-parantez y kapa-parantez eksi z

AnaDegiskenler
YeniDegiskenler
Sonuc