Program 4

New Attempt

Due Dec 7, 2022 by 11:59pm **Points** 10 **Submitting** a text entry box or a file upload

Program Specification:

Build a hash table using chaining as the collision resolution technique. Insertions into the hash table will correspond to declarations of variables and values in a program, searches will be requests for the value of a variable. Some variables will be local and have a narrow scope while some variables will be global.

The program will take input from a file, another program written in the omnipotent programming language BORG (Bionicly Omnipotent Resistance Grinders) and generate output from this program.

The BORG language has the following commands (keywords):

1. START-FINISH blocks. Indicating different scopes.
2. COM - Single line comments: Text should be ignored if on the same line
3. VAR varName – Variable Declaration, adds “varName” to the hash table.
4. variable = expression – Assignment statements, ie GEORGE = 122. Find GEORGE in the hash table and assign 122 to it.
5. ++ - increment operator, syntax: VARIABLE ++
6. - - decrement operator, syntax: VARIABLE --
7. expressions, expressions are limited to unary and binary arithmetic, or variable names
8. supported operators: + - / * % ^ (plus, minus, divide, multiple, modulo, exponent)
9. PRINT – syntax PRINT expression. If the expression is a variable, and this variable is not in scope, then an error message indicating unknown variable x at line number y. The value printed if there is a variable in scope should be the variable with the closest scope.
10. Errors – other than the print statements, our interpreter will not be responsible for detecting errors, syntax errors should be disregarded if encountered, assume that the source file is correct.

Our hash function: sum the ordinal values of the characters of the variable multiplied by their position in the string (1-indexing), then taking the modulo by TABLESIZE.

1. The variable ABC = (65 * 1 + 66 * 2 + 67 * 3) % TABLESIZE

All tokens are separated by one space or a new line.

Output: for this assignment, run your interpreter on this sample source program as well as a program of your own, and turn it the output from both, as well as the source code from your BORG program as well as source code of the assignment and its executable.

Example Submission:
BorgInterpretor.cpp
HelloWorld.txt
output.txt (or as a comment in the cpp file)

X-Credit. Each student may implement one additional feature to the language, such as adding if, switch, for, while, methods, more capable print statements. Only one student may implement a given extension to the language, and each extension must first be cleared with me.

Sample program and its output:

Input

```
COM HERE IS OUR FIRST BORG PROGRAM

COM WHAT A ROBUST LANGUAGE IT IS

START

  VAR BORAMIR = 25

  VAR LEGOLAS = 101
```

Output

```
PRINT BORAMIR                    BORAMIR IS 25

BORAMIR ++

PRINT LEGOLAS                    LEGOLAS IS 101

PRINT GANDALF                    GANDALF IS UNDEFINED

PRINT BORAMIR * 2                BOARAMIR * 2 IS 52

COM

COM NESTED BLOCK

COM

START

  VAR GANDALF = 49

  PRINT GANDALF                  GANDALF IS 49

  PRINT BORAMIR                  BORAMIR IS 26

FINISH

PRINT GANDALF                    GANDALF IS UNDEFINED
```

```
START

  LEGOLAS = 1000

  PRINT LEGOLAS                    LEGOLAS IS 1000

FINISH

PRINT LEGOLAS                      LEGOLAS IS 1000

LEGOLAS --

PRINT LEGOLAS                      LEGOLAS IS 999

FINISH
```