SE106, Fall 2012
Lab 3: Minimal BASIC Interpreter
Assigned: November 4
Due: December 2, 24:00

Introduction

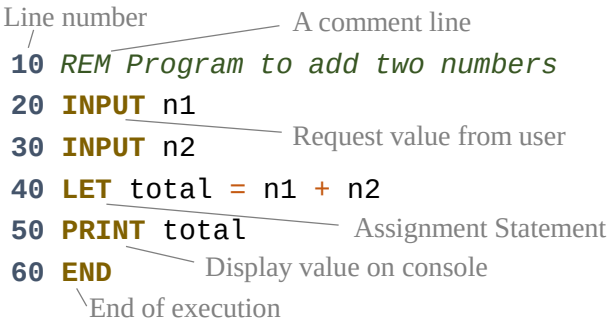
In this assignment, your mission is to build a minimal BASIC interpreter. You may start with the code for the expression evaluator in Computer Programming I (SE105).

BASIC Language and Interpreter

The programming language BASIC - the name is an acronym for Beginner's All-purpose Symbolic Instruction Code - was developed in the mid-1960s at Dartmouth College by John Kemeny and Thomas Kurtz.

A Plus B

In BASIC, a program consists of a sequence of numbered statements, as illustrated by the simple program below:



Lexical

Identifiers are formed by one or more letters. Keywords that are reserved words in the language and

cannot be used as identifiers. Integer literals are composed of digits only. This language is case-insensitive, which means `i` and `I` are the same variable, and `IF`, `if`, `If`, and even `iF` have the same meaning.

Line Numbers

The **line numbers *at the beginning of the line*** establish the sequence of operations in a program. In the absence of any control statements to the contrary, the statements in a program are **executed in ascending numerical order starting at the lowest number**.

Line numbers are also used to provide a simple editing mechanism. **Statements need not be entered in order**, because the line numbers indicate their relative position. Moreover, as long as the user has left gaps in the number sequence, new statements can be added in between other statements

For example, to change the program that adds two numbers into one that adds three numbers, you would need to make the following changes:

```
35 INPUT n3
40 LET total = n1 + n2 + n3
```

Add a new line, inserted into the program between line 30 and line 40

Replace the old line 40

The standard mechanism for deleting lines was to **type in a line number with nothing after it** on the line. Note that this operation actually **deleted the line** and did not simply replace it with a blank line that would appear in program listings.

Sequential Statements

REM	This statement is used for comments.
LET *var* = *exp*	This statement is BASIC's assignment statement.
PRINT *exp*	This statement print the value of the expression on the console and **then print a newline character**.
INPUT *var*	This statement print a **prompt consisting of the string " ? "** and then to read in a value to be stored in the variable.
END	Marks the end of the program. Execution halts when this line is reached. Execution **also stops if the program continues past the last numbered line**.

Control Statements

For example, the following BASIC program simulates a countdown from 10 to 0:

```
10 REM Program to simulate a countdown
20 LET T = 10
30 IF T < 0 THEN 70
40 PRINT T
50 LET T = T - 1
60 GOTO 30
70 END
```

Jumps to line 70 if the result of comparison is true

Jumps to line 30 unconditionally

GOTO *n*	This statement transfers control unconditionally to line *n* in the program. If line *n* **does not exist**, your BASIC interpreter should **generate an error message** informing the user of that fact.
IF *exp cmp exp* **THEN** *n*	This statement performs a conditional transfer of control. On encountering such a statement, the BASIC interpreter begins by evaluating condition, which in the minimal version of BASIC consists of two arithmetic expressions joined by one of the operators <, >, or =. If the result of the comparison is true, control passes to line n, **just as in the GOTO statement**; if not, the program continues with the next line in sequence.

Expressions

Expressions are used in LET, PRINT, and IF statements.

int_const	The simplest expressions are variables and integer constants.
var	
(*exp* **)**	These may be combined into larger expressions by enclosing an expression in parentheses or by joining two expressions with the operators +, -, *, and /, just as in the interpreter presented in the reader.
exp op exp	

Executed Directly

The LET, PRINT, and INPUT statements can be executed directly by typing them without a line number, in which case they are evaluated immediately. Thus, if you type in "PRINT 2 + 2" your program should respond immediately with 4.

The statements GOTO, IF, REM, and END are legal only if they appear as part of a program, which means that they must be given a line number.

BASIC Interpreter

These commands control the BASIC interpreter, which don't contained in BASIC program.

RUN	This command starts program execution **beginning at the lowest-numbered line**.
LIST	This command lists the steps in the program **in numerical sequence**.
CLEAR	This command deletes all program and variables.
QUIT	This command exits from the BASIC interpreter by calling `exit(0)`.
HELP	This command provides a simple help message describing your interpreter.

Error Reporting

DIVIDE BY ZERO	Calculating some number divide by zero.
NVALID NUMBER	User types wrong value to answer INPUT statement.
VARIABLE NOT DEFINED	A variable used before assigned it.
LINE NUMBER ERROR	GOTO statement's line number not exist.
SYNTAX ERROR	Any other errors.

Grading

Your implementation will be evaluated using `score` in `Test` folder. You can evaluate your implementation by yourself. Try "`./score -f –c`" to evaluate your program. Type "`./score –h`" for more details about our `score` program.

Hand-in

Your code must be written in C/C++. And, you are not allowed to use compiler-compiler or any other tools to generate your codes. You should work in `Basic` folder. You may add or modify files in this folder. Only keep your `Makefile` can produce executable file named `Basic` correctly when we type `make` in this folder. You should turn in your `Basic` folder via following command:

```
$ turnin lab3@cplusplus Basic
```

Remember to run `make clean` before your turn-in. Do NOT turn in `StanfordCPPLib` and executable files.