Lab 3: Minimal BASIC Interpreter

Assigned: November 6
Due: December 5, 23:59:59

Yunchao Wang(wangyunchao@sjtu.edu.cn) is responsible for this lab.

1 Environment

You must do this lab on Linux. (both windows and Mac was not supported)
And you can download all the resource from Pan.Baidu. Password: 0oyd

1.1 Get a Linux Environment

Actually, this lab work under most Linux OS with a not too old version.
So, if you already have a Linux Environment, just skip this section.

1.1.1 Install VM Management Platform

For windows, you can install VMware Workstation. You can get the package from Pan.Baidu or from website.
For Mac OS, you can install VMware Fusion. You can get the package from Pan.Baidu or from website.
And you can get Free Licence from SJTU Website.

1.1.2 Open the VM provide by TA

Download the three file in Pan.Baidu under "Virtul Machine" folder.
Open VMware Workstation or Fusion, and Click "File → Open", then choose "Ubuntu_For_SE232.ovf", and wait until import finish.

1.2 Start Lab in VM

The Password for VM is 123456.
You can use "VMware tools" to help you transmit file between VM and Host.
You can use "tar -zxvf linux.tar.gz" to decompress the souce code.

2 Introduction

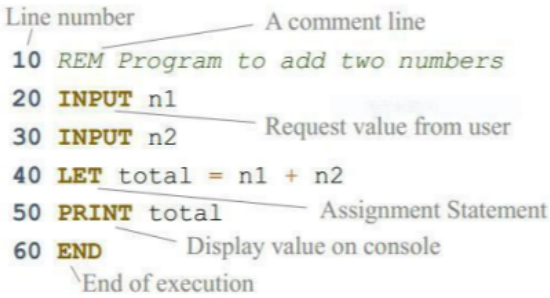
In this assignment, your mission is to build a minimal BASIC interpreter. You may start with the code TA provided.

3 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.

3.1 A Plus B

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

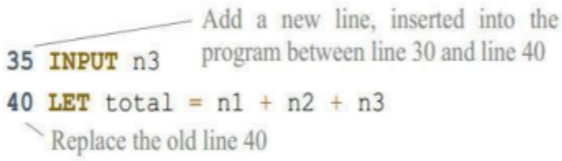


3.2 Lexical

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. **In this lab, the language is case-sensitive**, which means i and I are different variables, and IF, if, If, and even iF have different meanings.

3.3 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:



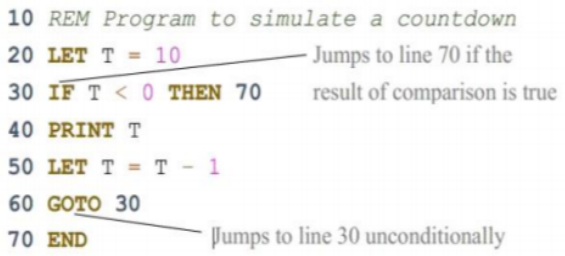
The standard mechanism for deleting lines was to **type in a line number with nothing after it** on the line. Note that this operation actually **deleted the line** and did not simply replace it with a blank line that would appear in program listings.

3.4 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**.

3.5 Control Statements

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



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.

3.6 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	

3.7 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.

3.8 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.

3.9 Error Reporting

DIVIDE BY ZERO	Calculating some number divide by zero.
INVALID 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.

4 Evaluation

Your score will be computed out of a maximum of 100 points based on the following rules:
Correctness: 100 trace files at 1 point each.
Style points: We expect you to have good comments, good OOP design and good code style. If your code smells terrible. You will lose up to 10 point.
Your solution will be tested for correctness on a Linux machine, using the same score program and trace files that were included in your lab directory. Your interpreter should produce identical output on these traces as the demo.
Cheaters will receive 0 point.

5 The score program

You can evaluate your implementation by yourself. Try "./score -f" under Basic folder to evaluate your program. Type "./score –h" for more details about our score program.

6 Hand-in

Your code must be written in 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. For Linux user, keep your Makefile can produce executable file named Basic correctly when we type make in this folder.
You should use the "handin" script to genterate your submit package, you can use it as flow:

```
./handin 51703XXXXXX_ZhangSan
```

and it will genterate a tar.gz file like "51703XXXXXX_ZhangSan.tar.gz", then, upload the tar.gz file onto the website.
Otherwise, you will lose your point.