2004/2005 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 4
Swamptran

To celebrate the 40th anniversary of interactive computing at Swamp County College your team is to recreate the first usable version of Swamptran (0.9). Swamptran is an interactive desk calculator with stored programs.

In the syntax description non-terminals are in *UPPERCASE ITALIC*. Within the right hand side of a definition elements are separated by ',' and the rule is terminated by '.'. Alternate choices are separated by '|'. Terminals are either quoted or described by prose enclosed in '(*' '*)'. [x] means that the enclosed element or elements are optional (may appear 0 or 1 times), [x]* means 0 or more times, [x]+ means 1 or more times. Parentheses '(' ')' are used for grouping.

All variables are global and are a single upper or lower case alphabetic character. All variables are integers in the range [-2000000000..2000000000]. Accessing a variable before it has been defined in a SET statement is a run time error.

VAR = *ID* .
ID = *ALPHA* .
ALPHA = (* an alpha character a-zA-Z, case is significant *) .
CONST = [*DIGIT*]+ (* A constant has the same range as a positive variable *) .
EXPR = [*VAR* | *CONST* | *BINOP* | *UNOP*]+ .
BINOP = '+' | '-' | '*' | '/' | '<' | '<=' | '>' | '>=' | '=' | '#=' | '&' | '|' .
UNOP = '!' | '#' .

Expressions are RPN (postfix). The stack has a depth of 100 and is cleared at the start of each expression. The value of an expression is the value left at the top of the stack. A variable or constant is just pushed onto the stack. With S0 as the top of the stack and S1 the second operand on the stack, the effect of the operators is:

The effect of *BINOP*: (... S1 S0) ⇒ (... (S1 *BINOP* S0))
The effect of *UNOP*: (... S0) ⇒ (... (*UNOP* S0))

'+', '-', '*', '/' are the usual binary arithmetic ops. '<', '<=', '>', '>=', '=', '#=' are logical comparisons ('#=' is not equal) resulting in 1 if true, 0 if false. '&', '|' are logical and, or. '#' is logical not. The logical operators convert an operand to 1 if non-zero and then perform their operation. '!' is unary minus. The results of an expression resulting in integer underflow or overflow are undefined.

Within an expression, variables and constants must be separated from other variables or constants by at least one blank. Ops must be separated from other ops by at least one blank.

A Swamptran session consists of a series of lines entered by the user and the response of the Swamptran processor.

LINE = *DIRECT* | *INDIRECT* .
INDIRECT = *STEP*, [*IFPART*], *CMD* .
DIRECT = [*IFPART*], *CMD* .
STEP = *CONST* (* in the range [1..32767] *) .

A *STEP* must be separated from what follows by at least one blank. A *LINE* may have leading blanks. A direct command is scanned for syntax and, if correct, executed. An indirect command is scanned for syntax and, if correct, stored for later execution, replacing any indirect command with the same step number.

IFPART = 'if', '(', *EXPR*, ')' .

Case is not significant. if, IF, iF, If are the same. During execution, if the *EXPR* of the optional *IFPART* is non-zero, the *CMD* is executed, otherwise it is not.

Problem 4
Swamptran (continued)

CMD = (‘goto’, *EXPR*) | ‘done’ | (‘print’, *EXPR*) | (‘printstep’, *EXPR*) | ‘printsteps’ | (‘set’, *VAR*, *EXPR*) | ‘clear’ | ‘reset’ | (‘deletestep’, *EXPR*) .

Case is not significant in the command keyword. goto, GOTO, gOTo, etc. are the same keywords. If the command requires an argument, there must be at least one blank after the command. The set command requires at least one blank between the variable and the expression.

direct commands:

goto: start execution of the stored program at the step *EXPR* and continue sequentially until a ‘done’ command, the highest numbered step has been executed, or a run time error occurs.

done: a no-op.

indirect commands:

goto: transfer control to step *EXPR*.

done: terminate execution of the program and accept input from the user.

direct or indirect commands:

print: print the value of the expression on a new line, with a ‘-’ if needed and no leading or trailing spaces or leading zeros.

printstep: print the step exactly as it was entered.

printsteps: print all steps exactly as entered.

set: define or redefine the variable with the value of the expression.

clear: undefine all variables.

reset: undefine all variable and erase all steps.

deletestep: erase the step.

syntax errors:

print the input line followed by the line:

```
eh?
```

note that

```
print
```

is a syntax error, while

```
print 1 +
```

will generate a run time error (stack underflow).

run time errors (n and m are step numbers, x is a variable, and a direct statement has step number 0):

error in step n step m undefined

error in step n undefined variable: x

error in step n divide by 0

error in step n stack underflow

description continued on next page

Problem 4
Swamptran (still continued)

Input is a series of lines terminated by end-of-file. A line may be at most 80 characters long. Output is the ouput of the Swamptran session.

Sample Input

```
100 set A a
200   set a a 1 -
300   if(a 1 =) done
400   set A A a *
500 goto 200
set a 5
goto 100
print A
printsteps
reset
printsteps
print a
print ab
print -1
print 1 !
```

Output for the Sample Input

```
120
100 set A a
200   set a a 1 -
300   if(a 1 =) done
400   set A A a *
500 goto 200
error in step 0 undefined variable: a
print ab
eh?
error in step 0 stack underflow
-1
```