A. Winsor

not clear, meaning; not simple-minded enough

79-14-2

NOTES ON THE SUBROUTINE & FOR LOOP PUSH-DOWN/POP-UP STACK. by A. Winsor

Subroutine levels and loop parameters are placed into the stack in a First-In-Last-Out (FILO) basis. Only those FOR Loops orginated at a particular subroutine level are available in that now level. Example that doesn't work:

```
    10 For A=1TO 5                Initialize Loop parameters.   define
    20 GOSUB 100                  Loop parameters get put into stack
     .
     .                            any code
     .
   100 NEXT A                     Requesting a test of the Loop that
                                  was orginated in a different sub level.
                                  The computer doesn't have this info
                                  in the current now level, but it is in
                                  the stack.
```

The computer cannot find any reference to the loop you are trying to test and breaks out of your program with a "WHAT?".

When a previous level of subroutine is popped up, current Loop parameters are lost and the previously stored loop parameters are available again. define

There are 20 levels available of subroutines. The 'now' level currently available to the computer and 19 levels in the stack. Try this short program to see:

```
  10 X=0              At the main now level with no subroutines.
  20 GOSUB 30         Pushing the current now level into the stack.
  30 X=X+1;PRINT X    Showing the subroutines in the stack.
  40 GOTO 20          Back to 20 to try another subroutine level.
```

The computer only lets you push 19 levels of subs into the stack. When the 20th level is tried to be pushed, the computer responds again with a "WHAT?".

When only Loops, or Loops & subroutine levels, are involved there are fewer spaces available in the stack. Example of only Loops:

```
  10 X="A"                 X=numeric value of "A"
  20 FOR A=1TO 5;PRINT "A"  Start the 'A' Loop and Print the letter
                            'A'. (Later will print B,C,D,E,etc.)
  30 X=X+1                  Increment X to the numeric value of 'B'
                            (later to C,D,E,etc.)
  40 %(-24566)=X*256+"FOR"  Poke in 'FOR A' into text line 20.
  50 %(-24558)=X*256+34     Poke in '"A' into text of line 20.
  60 GOTO 20                Go back to 20 to start next loop and
                            print out what loop was started.
```

Notice it only lets you have 16 Loop parameters (up to 'P').

When Loops and subroutine levels are mixed, the total allowed is usually about 17.

Code is not needed to terminate a loop if the loop parameters are never tested against before the loop is reinitialized again.

```
   10 FOR A=1TO 10;FOR B=1TO 5     Better to change line 20 to read
   20 IF @(A)=0A=10;B=5            20 IF @(A)=0 GOTO 100
   30 NEXT B;NEXT A;GOTO 100       When the Loops are started again
  100 do something;GOTO 10         the previous parameters are lost
                                   and so it doesn't matter you didn't
                                   finish it before.
```