1

COMPONENTS OF THE LANGUAGE
HOW MANY POTS OF PAINT DO YOU NEED TO PAINT THE ROOF AND WALL OF THIS WATER TANK?

WE COULD GO STRAIGHT AT IT LIKE THIS:

roof area, \( T = \pi \times (d/2)^2 = 33.2 \)

wall area, \( S = \pi \times d \times h = 551 \)

Total area, \( A = T + S = 884.2 \)

number of pots, \( G = A / 236 = 3.48 \)

rounding up, \( R = 3 \)

\( \therefore \) you need 3 pots of paint

OR WE COULD WRITE A PROGRAM (IN ENGLISH) TO SOLVE THE PROBLEM.

1. REMARK: A PROGRAM IN ENGLISH

2. THE FOLLOWING NUMBERS ARE THE DATA 6.5, 27, 236

3. READ THE DATA, CALLING THEM \( D, H, C \) RESPECTIVELY (think of this as putting the data into little boxes labelled \( D, H, C \) respectively - see opposite page)

4. WORK OUT 3.14 \( \times D^2 / 4 \) AND LET THE RESULT BE CALLED \( T \)

5. WORK OUT 3.14 \( \times D \times H \) AND LET THE RESULT BE CALLED \( S \)

6. ADD \( T \) TO \( S \) AND LET THE RESULT BE CALLED \( A \)

7. WORK OUT \( A + C \) AND LET THE RESULT BE CALLED \( G \)

8. ROUND \( G \) TO THE NEXT WHOLE NUMBER AND LET THE RESULT BE CALLED \( R \)

9. PRINT "YOU NEED" ; \( R \) ; " POTS"

(i.e. print whatever whole number \( R \) turns out to be)

10. THE END

THIS HAS THE ADVANTAGE OF BEING GOOD FOR ANY SIZE OF TANK AND PAINT POT \( \rightarrow \) YOU NEED ONLY REPLACE THE DATA ON LINE 2.
NOW

TRY OBEYING THE ENGLISH PROGRAM
OPPOSITE & FEEL WHAT IT WOULD BE LIKE
TO BE A COMPUTER & DEFILE THIS PAGE BY
WRITING NUMBERS IN THE LITTLE BOXES BELOW.

Here is the same program again
but written in BASIC.

Compare it carefully with the
English version opposite.

```
1 REM A PROGRAM IN BASIC
2 DATA 6.5, 27, 236
3 READ D, H, C
4 LET T = 3.14*D + 2/4
5 LET S = 3.14*D + H
6 LET A = T + S
7 LET G = A/C
8 LET R = INT (G + 1)
9 PRINT "YOU NEED" ; R ; "POTS"
10 END
```

You need 3 pots
PREPARE YOUR PROGRAM BY TYPING INSTRUCTIONS AT THE KEYBOARD. THE COMPUTER SIMPLY STORES THE PROGRAM AT THIS STAGE.

IT DOESN'T OBEY ANY INSTRUCTIONS.

1. REM A PROGRAM IN BASIC
2. DATA 6.5, 27, 236
3. READ D, H, C
4. LET T=3.14*D12/4
5. LET S+3.14*D+H
6. LET A+T+S
7. LET G=A/C
8. LET R=INT(G+1)
9. PRINT "YOU NEED";R;"POTS"
10. END

WHICH SETS THE COMPUTER TO WORK OBEYING THE STORED INSTRUCTIONS ONE AFTER THE OTHER IN NUMBERED SEQUENCE. EVENTUALLY THE COMPUTER WILL OBEY THE INSTRUCTION END THAT MAKES IT STOP.

RUN
YOU NEED 3 POTS

BEFORE YOU CAN TAKE THE FIRST STEP AND START TYPING THE PROGRAM YOU HAVE TO GO THROUGH THE RITUAL OF SIGNING ON AND TELLING THE COMPUTER YOU WANT TO USE BASIC.

DIFFERENT COMPUTERS (EVEN IDENTICAL COMPUTERS RUN BY DIFFERENT ORGANISATIONS) OFTEN HAVE DIFFERENT WAYS OF DOING THESE THINGS, SO IF YOU WANT TO TRY THE PROGRAM NOW GET SOMEONE WHO "KNOWS THE SYSTEM" TO SIGN ON FOR YOU AND CALL UP BASIC.
KEYBOARD

Every program in BASIC has to be typed on a keyboard

Probably something like this

Although positions of letters & digits are the same on most keyboards, keys like in the picture above vary in name, position and function from one installation to another.

Notice all letters are capital letters. Notice also there is a key for 1 and a key for zero (both in the top row). Never press the letters I and O in their place.

As on an ordinary typewriter, pressing at the same time as another key gives the character shown on the upper half of that key: thus together with gives # whereas alone, 3.

The "back arrow" serves to delete the character on its left from the computer's memory; two of them delete the previous two characters, and so on. Thus if you type PRIMPUNT then BASIC receives the word PRINT.

(remember this by exclaiming oh shift! when you hit the wrong key.) Some BASICS, however, use an underscore character for this purpose: PRIM...NT.

Most BASICS use a key (perhaps "rubout") which, when pressed, deletes the whole of the line you are typing from the computer's memory; another (perhaps "break") stops a program running.

For a new line in BASIC press

When typing "off line", press when typing "off line" press return then linefeed
IF YOU INTEND TO USE BASIC A LOT, LEARN TOUCH TYPING. TEN FINGERS ARE FASTER AND LESS FRUSTRATING THAN TWO.

THERE IS A LIMIT TO THE LENGTH OF A TYPED LINE. MOST BASICS ALLOW LINES UP TO 72 CHARACTERS LONG. SOME ALLOW LONGER LINES BUT IT IS BEST TO ACCEPT A LIMIT OF 72.

SOME BASICS ALLOW GREAT FREEDOM WITH THE SPACE BAR; SOME DISREGARD SPACES EXCEPT THOSE BETWEEN QUOTATION MARKS. THUS IT WOULD BE ALLOWABLE TO TYPE:

```
8 FOR D = $ TO P
```

INSTEAD OF:

```
8FORD=STOP
```

BUT IT IS OBVIOUSLY SILLY TO OBSCURE THE MEANING OF THE PROGRAM IN ORDER TO SAVE A FEW TAPS ON THE SPACE BAR.

SOME BASICS REFUSE TO ALLOW SPACES WITHIN THE CONTROLLING WORDS OF THE LANGUAGE. THUS THE FOLLOWING WOULD BE WRONG:

```
23 LET A = B + C
```

SOME BASICS DEMAND AT LEAST ONE SPACE BEFORE EACH CONTROLLING WORD, OR AFTER IT, OR BOTH:

```
20 DATA 6.5, 27, 236
80 PRINT "YOU NEED";'R;'"POTS"
```

SOME BASICS REFUSE TO ACCEPT SPACES WITHIN LINE NUMBERS BUT DO NOT OBJECT TO THEM INSIDE OTHER NUMBERS:

```
1000 LET A = 1 000 0
1000 LET A = 1000.0
```

SOME BASICS DO NOT ALLOW SPACES IN FRONT OF LINE NUMBERS:

```
95 LET A = B
100 LET C = D * F + G
```

GENERALLY WHEN ONE SPACE IS ALLOWED (OR DEMANDED) THEN SEVERAL ARE ALLOWED; AND GENERALLY A SPACE IS OPTIONAL ON EITHER SIDE OF THESE (, ; + / - = > < ) BUT NOT IN 1.5E2 (SEE PAGE 9) NOR BETWEEN > AND = (SEE PAGE 41).

A PROGRAM WHICH ACCEPTS ALL THESE RESTRICTIONS SHOULD BE ACCEPTABLE TO ANY VERSION OF BASIC.
LINE NUMBERS

10 REM A PROGRAM IN BASIC
20 DATA 6.5, 27, 286++++36
30 READ D, H, C
40 LET T = 3.14 * D/2
50 LET S = 3.14 * D+H
60 LET A = T+S
70 LET G = A/C
80 PRINT "YOU NEED" ; R ; "POTS"
90 END

There is a mistake in this program. The last LET was forgotten. Inserting it is simple: just type:

75 LET R = INT (G+1)

And the computer puts line 75 between line 70 & line 80. It makes no difference in what order you type the lines. The computer sorts them into ascending order of line number.

If you type several lines with the same line number the computer obliterates each previous version thus accepting the line typed last. If the line typed last is just a line number with nothing after it then the whole line vanishes from the computer's memory including the line number. That is how to delete unwanted lines. Thus:

120 LET A = B + C
125 LET E = F
120 LET A = B + G
125
120 LET A = B

Results in the computer remembering only:

120 LET A = B

The first line number in a program must be greater than 0. There is always a limit to the highest line number. Some BASICs stop at 9999, so it is best to accept this as the limit.

The last statement of every program must be:

120 LET A = B

(No other statement but the last may say END).
A **BASIC** PROGRAM IS A SEQUENCE OF NUMBERED LINES CALLED STATEMENTS.

A STATEMENT MAY SIMPLY STATE SOMETHING

30 READ A, B, C
40 LET G = A * B + 2 + C
50 PRINT "ANSWER IS"; G

OR IT MAY INSTRUCT THE COMPUTER TO DO SOMETHING.

A COMMON SYNONYM FOR STATEMENT IS INSTRUCTION.

THE STATEMENTS THAT DO THINGS ARE EXECUTABLE INSTRUCTIONS.

THE COMPUTER FINDS OUT WHAT IS STATED OR WHAT TO DO BY LOOKING AT THE FIRST WORDS: DATA, END, READ, LET, etc.

OR SOMETIMES AT THE FIRST TWO WORDS: MAT READ, MAT PRINT, etc. (WE MEET MAT ON PAGE 76).

BUT THERE IS AN IMPORTANT EXCEPTION!

THE WORD **LET** MAY BE OMITTED IN MOST VERSIONS OF BASIC.

40 G = A * B + 2 + C

**REM** STANDS FOR **REMARK**.

**REM** STATEMENTS CAUSE NO ACTION BY THE COMPUTER; YOU INCLUDE THEM TO CLARIFY YOUR PROGRAM.

10 REM *** WATER TANKS ***
20 REM
30 REM A PROGRAM TO ILLUSTRATE BASIC
40 REM ------------------------------
50 DATA 6.5, 27, 236
60 REM DIAM, HEIGHT, COVERAGE

THE EXAMPLES IN THIS BOOK DO NOT HAVE MANY "REM" STATEMENTS BECAUSE I HAVE ANNOTATED PROGRAMS WITH LITTLE ARROWS AND CLOUDS SO AS TO SAVE SPACE.
YOU CAN TYPE NUMBERS THREE WAYS AS INTEGERS, AS REALS OR IN E-FORM.

**INTEGER FORM** (WHOLE NUMBERS)

| 160 DATA 0, 2, +4, 1000, -30 |

**REAL FORM** (DECIMAL NUMBERS)

| 170 DATA +0.70, 4., -6, -L3, .98765 |

**E-FORM** (EXONENT FORM) WHERE E SAYS “TIMES TEN TO THE…”

| 190 DATA 1E3, 13.6E-4, -13.6E6, -0136E9 |

IN THE E-FORM THERE MUST ALWAYS BE A NUMBER IN FRONT OF THE E AND AN INTEGER AFTER IT. SOME BASICS ALLOW SPACES WITHIN AN E-FORM BUT IT IS BEST NOT TO HAVE THEM.

±10^{38}

IN SOME BASICS THE BIGGEST NUMBER THAT CAN BE STORED IS APPROXIMATELY ±10^{38} (BIG MEANS FAR FROM ZERO ON EITHER SIDE; SMALL MEANS CLOSE TO ZERO ON EITHER SIDE).

OTHER BASICS CAN HANDLE MUCH BIGGER NUMBERS THAN ±10^{38}; IT DEPENDS ON THE COMPUTER’S “WORD LENGTH” AND WHETHER THE “WORDS” ARE USED SINGLY, IN PAIRS, OR IN MULTIPLES. BUT NO BASIC SHOULD REFUSE TO HANDLE A NUMBER AS BIG AS ±100,000,000,000,000,000,000,000,000,000,000,000,000,000.

6−7 SIGNIFICANT DECIMAL DIGITS ≈ 987,654,321 WOULD BE STORED AS APPROXIMATELY 987,654,000. OTHER BASICS OFFER MUCH HIGHER PRECISION, 15 SIGNIFICANT FIGURES BEING TYPICAL. AGAIN IT DEPENDS ON THE COMPUTER’S “WORD LENGTH” AND HOW THE “WORDS” ARE USED. BUT NO BASIC SHOULD WORK TO LESS PRECISION THAN 6-TO-7 SIG. FIGS. (THE VAGUENESS OF “6 TO 7” IS BECAUSE MOST COMPUTERS USE BINARY ARITHMETIC, NOT DECIMAL.) A MORE PRECISE RENDERING WOULD BE “24 BINARY DIGITS FOR POSITIVE NUMBERS; 23 FOR NEGATIVE; OR VICE VERSA” BUT THESE IMPLICATIONS NEED NOT BOther THE NOVICE TO BASIC. 
WE SAW SOME OF THESE ON PAGE 3. THEY ARE THE LITTLE BOXES USED TO HOLD NUMBERS.

THE FULL 286 ARE SHOWN BELOW. IT IS USEFUL TO KEEP A LARGE-SCALE CHART LIKE THIS AND MAKE A PHOTOCOPY FOR EACH NEW PROGRAM. AS YOU USE EACH VARIABLE, WRITE A NOTE IN ITS BOX SAYING WHAT YOU USE IT FOR. THIS STOPS YOU USING VARIABLES ALREADY USED FOR SOMETHING ELSE — A COMMON SOURCE OF TROUBLE.

<table>
<thead>
<tr>
<th>A</th>
<th>A0</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B0</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
<td>B4</td>
<td>B5</td>
<td>B6</td>
<td>B7</td>
<td>B8</td>
<td>B9</td>
</tr>
<tr>
<td>C</td>
<td>C0</td>
<td>C1</td>
<td>C2</td>
<td>C3</td>
<td>C4</td>
<td>C5</td>
<td>C6</td>
<td>C7</td>
<td>C8</td>
<td>C9</td>
</tr>
<tr>
<td>D</td>
<td>D0</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
</tr>
<tr>
<td>E</td>
<td>E0</td>
<td>E1</td>
<td>E2</td>
<td>E3</td>
<td>E4</td>
<td>E5</td>
<td>E6</td>
<td>E7</td>
<td>E8</td>
<td>E9</td>
</tr>
<tr>
<td>F</td>
<td>F0</td>
<td>F1</td>
<td>F2</td>
<td>F3</td>
<td>F4</td>
<td>F5</td>
<td>F6</td>
<td>F7</td>
<td>F8</td>
<td>F9</td>
</tr>
<tr>
<td>G</td>
<td>G0</td>
<td>G1</td>
<td>G2</td>
<td>G3</td>
<td>G4</td>
<td>G5</td>
<td>G6</td>
<td>G7</td>
<td>G8</td>
<td>G9</td>
</tr>
<tr>
<td>H</td>
<td>H0</td>
<td>H1</td>
<td>H2</td>
<td>H3</td>
<td>H4</td>
<td>H5</td>
<td>H6</td>
<td>H7</td>
<td>H8</td>
<td>H9</td>
</tr>
<tr>
<td>I</td>
<td>I0</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
<td>I6</td>
<td>I7</td>
<td>I8</td>
<td>I9</td>
</tr>
<tr>
<td>J</td>
<td>J0</td>
<td>J1</td>
<td>J2</td>
<td>J3</td>
<td>J4</td>
<td>J5</td>
<td>J6</td>
<td>J7</td>
<td>J8</td>
<td>J9</td>
</tr>
<tr>
<td>K</td>
<td>K0</td>
<td>K1</td>
<td>K2</td>
<td>K3</td>
<td>K4</td>
<td>K5</td>
<td>K6</td>
<td>K7</td>
<td>K8</td>
<td>K9</td>
</tr>
<tr>
<td>L</td>
<td>L0</td>
<td>L1</td>
<td>L2</td>
<td>L3</td>
<td>L4</td>
<td>L5</td>
<td>L6</td>
<td>L7</td>
<td>L8</td>
<td>L9</td>
</tr>
<tr>
<td>M</td>
<td>M0</td>
<td>M1</td>
<td>M2</td>
<td>M3</td>
<td>M4</td>
<td>M5</td>
<td>M6</td>
<td>M7</td>
<td>M8</td>
<td>M9</td>
</tr>
<tr>
<td>N</td>
<td>N0</td>
<td>N1</td>
<td>N2</td>
<td>N3</td>
<td>N4</td>
<td>N5</td>
<td>N6</td>
<td>N7</td>
<td>N8</td>
<td>N9</td>
</tr>
<tr>
<td>O</td>
<td>O0</td>
<td>O1</td>
<td>O2</td>
<td>O3</td>
<td>O4</td>
<td>O5</td>
<td>O6</td>
<td>O7</td>
<td>O8</td>
<td>O9</td>
</tr>
<tr>
<td>P</td>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td>P4</td>
<td>P5</td>
<td>P6</td>
<td>P7</td>
<td>P8</td>
<td>P9</td>
</tr>
<tr>
<td>Q</td>
<td>Q0</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
<td>Q5</td>
<td>Q6</td>
<td>Q7</td>
<td>Q8</td>
<td>Q9</td>
</tr>
<tr>
<td>R</td>
<td>R0</td>
<td>R1</td>
<td>R2</td>
<td>R3</td>
<td>R4</td>
<td>R5</td>
<td>R6</td>
<td>R7</td>
<td>R8</td>
<td>R9</td>
</tr>
<tr>
<td>S</td>
<td>S0</td>
<td>S1</td>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>S5</td>
<td>S6</td>
<td>S7</td>
<td>S8</td>
<td>S9</td>
</tr>
<tr>
<td>T</td>
<td>T0</td>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
<td>T5</td>
<td>T6</td>
<td>T7</td>
<td>T8</td>
<td>T9</td>
</tr>
<tr>
<td>U</td>
<td>U0</td>
<td>U1</td>
<td>U2</td>
<td>U3</td>
<td>U4</td>
<td>U5</td>
<td>U6</td>
<td>U7</td>
<td>U8</td>
<td>U9</td>
</tr>
<tr>
<td>V</td>
<td>V0</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
<td>V6</td>
<td>V7</td>
<td>V8</td>
<td>V9</td>
</tr>
<tr>
<td>W</td>
<td>W0</td>
<td>W1</td>
<td>W2</td>
<td>W3</td>
<td>W4</td>
<td>W5</td>
<td>W6</td>
<td>W7</td>
<td>W8</td>
<td>W9</td>
</tr>
<tr>
<td>X</td>
<td>X0</td>
<td>X1</td>
<td>X2</td>
<td>X3</td>
<td>X4</td>
<td>X5</td>
<td>X6</td>
<td>X7</td>
<td>X8</td>
<td>X9</td>
</tr>
<tr>
<td>Y</td>
<td>Y0</td>
<td>Y1</td>
<td>Y2</td>
<td>Y3</td>
<td>Y4</td>
<td>Y5</td>
<td>Y6</td>
<td>Y7</td>
<td>Y8</td>
<td>Y9</td>
</tr>
<tr>
<td>Z</td>
<td>Z0</td>
<td>Z1</td>
<td>Z2</td>
<td>Z3</td>
<td>Z4</td>
<td>Z5</td>
<td>Z6</td>
<td>Z7</td>
<td>Z8</td>
<td>Z9</td>
</tr>
</tbody>
</table>