**FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '88**
**BASIC PROGRAM SOLUTIONS**

```
'1.1
' This program clears the screen and prints a phrase
'
CLS
FOR I = 1 TO 10
  PRINT "THE BEST COMPUTER CONTEST!"
NEXT I


'1.2
' This program determines if a given input is integer or real.
'
INPUT "Enter #:"; NUM
IF NUM = INT(NUM) THEN PRINT "INTEGER" ELSE PRINT "REAL"


'1.3
' This program calculates the number of bytes on N diskettes.
'
INPUT "Enter N: "; N
PRINT N * 40 * 8 * 512


'1.4
' This program prints the computer component missing.
'
INPUT "Enter component:"; A$
INPUT "Enter component:"; B$
INPUT "Enter component:"; C$
INPUT "Enter component:"; D$
DATA CPU,PRIMARY,SECONDARY,INPUT,OUTPUT
FOR I = 1 TO 5
  READ E$
  IF NOT (A$ = E$ OR B$ = E$ OR C$ = E$ OR D$ = E$) THEN
    PRINT E$: END
  END IF
NEXT I
```

```
'1.5
' This program displays 4 rectangles of asterisks with #s.
'
CLS
FOR I = 1 TO 79: PRINT "*"; : NEXT I
FOR I = 2 TO 23
  LOCATE I, 1: PRINT "*"
  LOCATE I, 40: PRINT "*"
  LOCATE I, 79: PRINT "*"
NEXT I
LOCATE 12, 1: FOR I = 1 TO 79: PRINT "*"; : NEXT I
LOCATE 24, 1: FOR I = 1 TO 79: PRINT "*"; : NEXT I
' Place numbers in center of rectangles
LOCATE 6, 20: PRINT 1
LOCATE 6, 60: PRINT 2
LOCATE 18, 20: PRINT 3
LOCATE 18, 60: PRINT 4
```

```
'1.6
' This program displays the acronym for a given set of words.
'
INPUT "Enter words:"; A$
PRINT MID$(A$, 1, 1);
FOR I = 2 TO LEN(A$)
  MD$ = MID$(A$, I, 1)
  IF MD$ = " " THEN PRINT MID$(A$, I + 1, 1); : I = I + 1
NEXT I
```

```
'1.7
' This program will display 3 computer names in order of size.
'
INPUT "Enter name:"; N1$
INPUT "Enter type:"; T1$
INPUT "Enter name:"; N2$
INPUT "Enter type:"; T2$
INPUT "Enter name:"; N3$
INPUT "Enter type:"; T3$
IF T1$ = "MICRO" THEN PRINT N1$
IF T2$ = "MICRO" THEN PRINT N2$
IF T3$ = "MICRO" THEN PRINT N3$
IF T1$ = "MINI" THEN PRINT N1$
IF T2$ = "MINI" THEN PRINT N2$
IF T3$ = "MINI" THEN PRINT N3$
IF T1$ = "MAINFRAME" THEN PRINT N1$
IF T2$ = "MAINFRAME" THEN PRINT N2$
IF T3$ = "MAINFRAME" THEN PRINT N3$
```

```
'1.8
' This program will count the number of cans to be stacked.
'
INPUT "Enter N: "; N
FOR I = N TO 1 STEP -2
  SUM = SUM + I
NEXT I
PRINT SUM
```

```
'1.9
' This program simulates a queue w/options: ADD, TAKE, QUIT.
'
INPUT "Enter command:"; IN$
WHILE IN$ <> "QUIT"
  IF IN$ = "ADD" THEN
    MAX = MAX + 1: INPUT "Enter integer:"; A(MAX)
  ELSE
    IF IN$ = "TAKE" THEN
      MIN = MIN + 1: PRINT A(MIN)
    END IF
  END IF
  INPUT "Enter command:"; IN$
WEND
```

```
'1.10
' This program determines events of history between dates.
'
DATA 1642,"BLAISE PASCAL","ADDING MACHINE"
DATA 1801,"JOSEPH JACQUARD","PUNCHCARD AND WEAVING LOOM"
DATA 1830,"CHARLES BABBAGE","DESIGN OF ANALYTIC ENGINE"
DATA 1890,"HERMAN HOLLERITH","PUNCHCARD TABULATING MACHINE"
DATA 1944,"HOWARD AIKEN","MARK I"
DATA 1946,"ECKERT AND MAUCHLY","ENIAC"
DATA 1949,"VON NEUMAN","EDVAC"
'
INPUT "Enter years: "; Y1, Y2
FOR I = 1 TO 7
   READ DAT, NAM$, INV$
   IF Y1 <= DAT AND DAT <= Y2 THEN PRINT NAM$; " INVENTED "; INV$
NEXT I
```

```
'2.1
' This program displays a solid diamond of asterisks.
'
INPUT "Enter N: "; N
FOR I = 1 TO N STEP 2
  PRINT SPACE$((N - I) / 2);
  FOR J = 1 TO I: PRINT "*"; : NEXT J: PRINT
NEXT I
'
FOR I = N - 2 TO 1 STEP -2
  PRINT SPACE$((N - I) / 2);
  FOR J = 1 TO I: PRINT "*"; : NEXT J: PRINT
NEXT I


'2.2
' This program determines the efficiency order of 3 sorts.
'
INPUT "Enter N: "; N
B = N * (N - 1) / 2: B$ = "BUBBLE SORT"
S = N * (LOG(N) / LOG(2)) ^ 2: S$ = "SHELL SORT"
Q = N * (LOG(N) / LOG(2)): Q$ = "QUICK SORT"
'
IF B < S AND B < Q THEN
  PRINT B$
  IF S < Q THEN PRINT S$ ELSE PRINT Q$
  IF S < Q THEN PRINT Q$ ELSE PRINT S$
  END
ELSE
  IF S < Q THEN
    PRINT S$
    IF B < Q THEN PRINT B$ ELSE PRINT Q$
    IF B < Q THEN PRINT Q$ ELSE PRINT B$
    END
  ELSE
    PRINT Q$
    IF B < S THEN PRINT B$ ELSE PRINT S$
    IF B < S THEN PRINT S$ ELSE PRINT B$
  END IF
END IF
```

```
'2.3
' This program determines the number of people in a group.
'
DEFINT A-Z
DIV(1) = 2: RE(1) = 1
DIV(2) = 3: RE(2) = 2
DIV(3) = 5: RE(3) = 1
DIV(4) = 7: RE(4) = 2
'
FOR NUM = 1 TO 200
  GOOD = -1
  FOR I = 1 TO 4
    IF NUM MOD DIV(I) <> RE(I) THEN GOOD = 0
  NEXT I
  IF GOOD THEN PRINT NUM: END
NEXT NUM


'2.4
' This program generates 5 random numbers between 0 and 9999.
'
INPUT "Enter seed:"; SEED
FOR I = 1 TO 5
  PROD# = SEED * SEED
  PROD$ = MID$(STR$(PROD#), 2)
  DIGITS = LEN(PROD$)
  IF DIGITS < 8 THEN
'   **** Pad 0's to make 8 digit # ****
    FOR J = 1 TO 8 - DIGITS
      PROD$ = PROD$ + "0"
    NEXT J
  END IF
'
  SEED = VAL(MID$(PROD$, 3, 4))
  PRINT SEED
NEXT I


'2.5
' This program checks to see if data trasmitted is Correct.
'
INPUT "Enter bits:"; BIT$
INPUT "Enter parity:"; PAR$
IF LEN(BIT$) <> 8 THEN PRINT "ERROR": END
FOR I = 1 TO 8
  MD$ = MID$(BIT$, I, 1)
  IF MD$ <> "0" AND MD$ <> "1" THEN PRINT "ERROR": END
  SUM = SUM + VAL(MD$)
NEXT I
' ERROR if even but odd parity;  or if odd but even parity
IF SUM MOD 2 = 0 AND PAR$ <> "EVEN" THEN PRINT "ERROR": END
IF SUM MOD 2 = 1 AND PAR$ <> "ODD" THEN PRINT "ERROR": END
PRINT "CORRECT"
```

```
'2.6
' This program will calculate the area of a polygon.
'
INPUT "Enter n: "; N
FOR I = 1 TO N
  INPUT "Enter vertex: "; X(I), Y(I)
NEXT I
'
X(N + 1) = X(1): Y(N + 1) = Y(1)
FOR I = 1 TO N
  SUM = SUM + X(I) * Y(I + 1) - Y(I) * X(I + 1)
NEXT I
PRINT USING "AREA = ##.#"; ABS(SUM) / 2


'2.7
' This program displays the date before/after a given date.
'
INPUT "Enter month, day, year: "; MONTH, DAY, YEAR
DIM MO(12)
FOR I = 1 TO 12: READ MO(I): NEXT I
DATA 31,28,31,30,31,30,31,31,30,31,30,31
'
D1 = DAY - 1: D2 = DAY + 1: M1 = MONTH: M2 = MONTH
Y1 = YEAR: Y2 = YEAR
IF Y1 MOD 4 = 0 AND Y1 MOD 100 > 0 THEN LEAP = -1
IF LEAP AND M1 = 3 AND D1 = 0 THEN LEAP1 = 1
IF LEAP AND M2 = 2 AND D2 = 29 THEN LEAP2 = 1
'
IF D1 = 0 THEN
  M1 = M1 - 1
  IF M1 > 0 THEN D1 = MO(M1) + LEAP1
  IF M1 = 0 THEN M1 = 12: D1 = MO(M1): Y1 = Y1 - 1
ELSE
  IF D2 > MO(M2) + LEAP2 THEN
    M2 = M2 + 1: D2 = 1
    IF M2 > 12 THEN M2 = 1: Y2 = Y2 + 1
  END IF
END IF
'
PRINT LTRIM$(STR$(M1));
PRINT "-"; LTRIM$(STR$(D1)); "-"; LTRIM$(STR$(Y1))
PRINT LTRIM$(STR$(M2));
PRINT "-"; LTRIM$(STR$(D2)); "-"; LTRIM$(STR$(Y2))
```

```
'2.8
' This program displays a student's Cumulative G. P. Ave.
'
SEM = 1
WHILE SEM <= 8
  TOTAL = 0: HRSTOT = 0
  FOR I = 1 TO 4
    INPUT "Enter grade, credits:"; GR$, HRS
    POYNTS = 4 - (ASC(GR$) - 65)       'A=4 B=3 C=2 D=1 F=-1
    IF POYNTS = -1 THEN POYNTS = 0     'F=-1 becomes F=0
    TOTAL = TOTAL + POYNTS * HRS
    HRSTOT = HRSTOT + HRS
  NEXT I
'
  GPA = TOTAL / HRSTOT
  PRINT USING " GPA= #.###"; GPA
  CUMTOTAL = CUMTOTAL + TOTAL: CUMHRS = CUMHRS + HRSTOT
  CGPA = CUMTOTAL / CUMHRS
  PRINT USING "CGPA= #.###"; CGPA
  IF CGPA < 1 THEN DIS = -1
  IF CGPA < 2 AND LASTCGPA < 2 AND SEM > 1 THEN DIS = -1
  IF DIS THEN PRINT "STUDENT IS DISMISSED": END
  LASTCGPA = CGPA
  SEM = SEM + 1
WEND
```

```
'2.9
' This program displays 2 elements that form a battery.
'
DATA "LITHIUM ",+3.05
DATA "SODIUM  ",+2.71
DATA "ZINC    ",+0.76
DATA "IRON    ",+0.44
DATA "TIN     ",+0.14
DATA "IODINE  ",-0.54
DATA "SILVER  ",-0.80
DATA "MERCURY ",-0.85
DATA "BROMINE ",-1.09
DATA "CHLORINE",-1.36
FOR I = 1 TO 10: READ ELEM$(I), POT(I): NEXT I
'
INPUT "Enter Desired Voltage, Tolerance: "; VOLT, TOL
'
FOR I = 1 TO 10
  FOR J = 1 TO 10
    DIF = POT(I) - POT(J)
    IF DIF >= VOLT - TOL AND DIF <= VOLT + TOL THEN
      COUNT = COUNT + 1
      IF COUNT = 1 AND DISPLAY > 0 THEN
        PRINT "PRESS ANY KEY FOR MORE": A$ = INPUT$(1): PRINT
      END IF
      PRINT ELEM$(I); "   "; ELEM$(J); "   ";
      PRINT USING "#.##"; DIF
      DISPLAY = 1
    END IF
    IF COUNT = 8 THEN PRINT : COUNT = 0
  NEXT J
NEXT I
IF DISPLAY = 0 THEN PRINT "NO BATTERY CAN BE FORMED"
```

```
'2.10
' This program will keep score for a double dual race.
'
CLS : DIM IN$(21)
FOR I = 1 TO 21
  PRINT "Place "; I; ":"; : INPUT IN$(I)
  IF I > 1 THEN
    J = 1
    WHILE J <= TN AND INIT$(J) <> IN$(I): J = J + 1: WEND
  END IF
  IF (INIT$(J) <> IN$(I)) OR (I = 1) THEN
    TN = TN + 1: INIT$(TN) = IN$(I)
  END IF
NEXT I
'    Assert  TEAM$(1, 2, 3) = 3 unique team INITIALS
FOR I = 1 TO 2
  FOR J = I + 1 TO 3
    PL = 0: T1 = 0: T2 = 0: T1PL = 0: T2PL = 0
    FOR K = 1 TO 21
      IF IN$(K) = INIT$(I) THEN
     PL = PL + 1: T1 = T1 + PL: T1PL = T1PL + 1
     TEAM1(T1PL) = PL
      END IF
      IF IN$(K) = INIT$(J) THEN
     PL = PL + 1: T2 = T2 + PL: T2PL = T2PL + 1
     TEAM2(T2PL) = PL
      END IF
    NEXT K
    T1 = T1 - TEAM1(6) - TEAM1(7)
    T2 = T2 - TEAM2(6) - TEAM2(7)
    PRINT "TEAM "; INIT$(I); ":"; T1; " POINTS"
    PRINT "TEAM "; INIT$(J); ":"; T2; " POINTS"
    IF (T1 < T2) OR (T1 = T2 AND TEAM1(6) < TEAM2(6)) THEN
      PRINT "TEAM "; INIT$(I);
    ELSE
      PRINT "TEAM "; INIT$(J);
    END IF
    PRINT " WINS!": PRINT
  NEXT J
NEXT I
```

```
'3.1
' This program puts a set of real numbers in numerical order.
'
INPUT "Enter N: "; N
FOR I = 1 TO N
  INPUT "Enter #: "; A(I)
NEXT I
DATA 0,8,1,2,5,4,3,9,7,6
FOR I = 0 TO 9: READ PLACE: ORDER(PLACE) = I: NEXT I
'    ***  replace digits in duplicated number  ***
FOR I = 1 TO N
   NUM$ = STR$(A(I))
   FOR J = 1 TO LEN(NUM$)
     MD$ = MID$(NUM$, J, 1)
     NUM = VAL(MD$)
     IF NUM > 0 OR MD$ = "0" THEN
       NUM2 = ORDER(NUM)
       MID$(NUM$, J, 1) = MID$(STR$(NUM2), 2)
     END IF
  NEXT J
  B(I) = VAL(NUM$)
NEXT I
'   ***   sort according to numbers with replaced digits  ***
FOR I = 1 TO N - 1
  FOR J = I + 1 TO N
    IF B(I) > B(J) THEN SWAP B(I), B(J): SWAP A(I), A(J)
  NEXT J
NEXT I
FOR I = 1 TO N: PRINT LTRIM$(STR$(A(I))): NEXT I
```

```
'3.2
' This program displays total number of ways to make change.
'
DEFINT B-Z
INPUT "Enter AMOUNT: "; AMOUNT
MAXQ = INT(AMOUNT * 4)
MAXD = INT(AMOUNT * 10)
MAXN = INT(AMOUNT * 20)
FOR Q = 0 TO MAXQ
  FOR D = 0 TO MAXD - INT(2.5 * Q)
    FOR N = 0 TO MAXN - 5 * Q - 2 * D
      COUNT = COUNT + 1
    NEXT N
  NEXT D
NEXT Q
PRINT COUNT
```

```
'3.3
' This program determines if a point/box is inside a 2nd box.
'
INPUT "Enter point: "; PX, PY, PZ
INPUT "Enter cube1 diagonal point1: "; C1X1, C1Y1, C1Z1
INPUT "Enter cube1 diagonal point2: "; C1X2, C1Y2, C1Z2
INPUT "Enter cube2 diagonal point1: "; C2X1, C2Y1, C2Z1
INPUT "Enter cube2 diagonal point2: "; C2X2, C2Y2, C2Z2
A = C1X1: B = C1X2: GOSUB MinOfAandB: C1MINX = MIN
A = C1Y1: B = C1Y2: GOSUB MinOfAandB: C1MINY = MIN
A = C1Z1: B = C1Z2: GOSUB MinOfAandB: C1MINZ = MIN
A = C2X1: B = C2X2: GOSUB MinOfAandB: C2MINX = MIN
A = C2Y1: B = C2Y2: GOSUB MinOfAandB: C2MINY = MIN
A = C2Z1: B = C2Z2: GOSUB MinOfAandB: C2MINZ = MIN
A = C1X1: B = C1X2: GOSUB MaxOfAandB: C1MAXX = MAX
A = C1Y1: B = C1Y2: GOSUB MaxOfAandB: C1MAXY = MAX
A = C1Z1: B = C1Z2: GOSUB MaxOfAandB: C1MAXZ = MAX
A = C2X1: B = C2X2: GOSUB MaxOfAandB: C2MAXX = MAX
A = C2Y1: B = C2Y2: GOSUB MaxOfAandB: C2MAXY = MAX
A = C2Z1: B = C2Z2: GOSUB MaxOfAandB: C2MAXZ = MAX
'
PRINT "POINT ";
IF PX < C2MINX OR PY < C2MINY OR PZ < C2MINZ THEN
  PRINT "DOES NOT LIE";
ELSE
  IF PX > C2MAXX OR PY > C2MAXY OR PZ > C2MAXZ THEN
    PRINT "DOES NOT LIE";
  ELSE
    PRINT "LIES";
  END IF
END IF
PRINT " INSIDE 2ND CUBE"
'
PRINT "1ST CUBE ";
IF C1MINX < C2MINX OR C1MINY < C2MINY OR C1MINZ < C2MINZ THEN
  PRINT "DOES NOT LIE";
ELSE
  IF C1MAXX > C2MAXX OR C1MAXY > C2MAXY OR C1MAXZ > C2MAXZ THEN
    PRINT "DOES NOT LIE";
  ELSE
    PRINT "LIES";
  END IF
END IF
PRINT " INSIDE 2ND CUBE"
END
'***   SUBROUTINE to determine MIN of A and B
MinOfAandB:
  IF A <= B THEN MIN = A ELSE MIN = B
  RETURN
'***   SUBROUTINE to determine MAX of A and B
MaxOfAandB:
  IF A >= B THEN MAX = A ELSE MAX = B
  RETURN
```

```
'3.4
' This program produces an alphabetical list of permutations.
' **** Note: QBASIC has recursive capabilities, but this is
'            a way to do permutations without recursion.
' Also, this is an example of old BASIC (with line #s/branching).
'
10 DIM PERM$(720)
20 INPUT "Enter letters:"; A$: L = LEN(A$)
30 FOR I = 1 TO L: B$(I) = MID$(A$, I, 1): NEXT I: I = L
40 ON I GOTO 20, 90, 80, 70, 60, 50
50 FOR N6 = 1 TO 6: H = 5: GOSUB 340
60   FOR N5 = 1 TO 5: H = 4: GOSUB 340
70     FOR N4 = 1 TO 4: H = 3: GOSUB 340
80       FOR N3 = 1 TO 3: H = 2: GOSUB 340
90        FOR N2 = 1 TO 2
100          SWAP B$(I), B$(I - 1): TOTAL = TOTAL + 1
110          FOR J = 1 TO L
120            PERM$(TOTAL) = PERM$(TOTAL) + B$(J)
130          NEXT J
140        NEXT N2: IF I = 2 THEN 190
150       NEXT N3: IF I = 3 THEN 190
160     NEXT N4: IF I = 4 THEN 190
170   NEXT N5: IF I = 5 THEN 190
180 NEXT N6
190 '***  INSERTION SORT ***
200 FOR I = 2 TO TOTAL
210   IND = I
220   WHILE PERM$(IND) < PERM$(IND - 1) AND IND > 1
230     SWAP PERM$(IND), PERM$(IND - 1): IND = IND - 1
240   WEND
250 NEXT I
260 '
270 FOR I = 1 TO TOTAL                        'With QBASIC,
280   IF PERM$(I) = PERM$(I - 1) THEN 300     '<==This can be written
290   PRINT PERM$(I): TOTAL2 = TOTAL2 + 1     'using IF/END IF
300 NEXT I                                    'instead of branching.
310 PRINT "TOTAL="; TOTAL2
320 END
330 ' *****  SUBROUTINE *****
340 Z$ = B$(I - H)
350 FOR J = I - H TO I - 1
360   B$(J) = B$(J + 1)
370 NEXT J
380 B$(I) = Z$
390 RETURN
```

```
'3.5
' This program will control the movements of a snake.
'
CLS : DIM A(25, 81)
V = 12: H = 8: LOCATE V, H
FOR I = 8 TO 32
  PRINT "*"; : A(V, I) = 1
  A$ = A$ + "12":  B$ = B$ + RIGHT$(STR$(I), 2)
NEXT I
WHILE D$ = "": D$ = INKEY$: WEND: C$ = D$
DO UNTIL C$ = CHR$(27)
  FOR I = 1 TO 100
    D$ = INKEY$:  IF D$ <> "" THEN C$ = D$
  NEXT I
  IF C$ = "I" THEN V = V - 1
  IF C$ = "M" THEN V = V + 1
  IF C$ = "J" THEN H = H - 1
  IF C$ = "K" THEN H = H + 1
  IF A(V, H) OR V = 0 OR V = 25 OR H = 0 OR H = 81 THEN END
  A(V, H) = 1: LOCATE V, H: PRINT "*"
  X = VAL(RIGHT$(A$, 2)): Y = VAL(RIGHT$(B$, 2))
  LOCATE X, Y: PRINT " "
  A(X, Y) = 0
  A$ = LEFT$(A$, 24 * 2): B$ = LEFT$(B$, 24 * 2)
  A$ = RIGHT$(STR$(V), 2) + A$
  B$ = RIGHT$(STR$(H), 2) + B$
LOOP
```

```
'3.6
' This program will solve two linear equations.
'
INPUT "Enter equation 1: "; E1$
INPUT "Enter equation 2: "; E2$
'
' Determine coeficients A1,B1,C1 and A2,B2,C2
'
EQ$ = E1$: ST = 1: GOSUB ParseEq: : A1 = VAAL
EQ$ = E1$: GOSUB ParseEq: : B1 = VAAL
EQ$ = E1$: GOSUB ParseEq: : C1 = VAAL
EQ$ = E2$: ST = 1: GOSUB ParseEq: : A2 = VAAL
EQ$ = E2$: GOSUB ParseEq: : B2 = VAAL
EQ$ = E2$: GOSUB ParseEq: : C2 = VAAL
'
' Compute solution if it exists
'
DEN = A1 * B2 - A2 * B1
NUMX = C1 * B2 - C2 * B1
NUMY = A1 * C2 - A2 * C1
IF DEN = 0 THEN PRINT "NO UNIQUE SOLUTION EXISTS.": END
PRINT "XSOLUTION= ";
IF NUMX / DEN < 0 THEN
  PRINT USING "##.#"; NUMX / DEN;
ELSE
  PRINT USING "#.#"; NUMX / DEN;
END IF
PRINT "   YSOLUTION= ";
IF NUMY / DEN < 0 THEN
  PRINT USING "##.#"; NUMY / DEN
ELSE
  PRINT USING "#.#"; NUMY / DEN
END IF
END
'
' Find Starting position ST of value
'
ParseEq:
  SYGN = 1    'Default to 1 (positive for unsigned #s)
  MD$ = "="
  WHILE MD$ = "="
    MD$ = MID$(EQ$, ST, 1)
     IF MD$ = "X" THEN VAAL = 1: ST = ST + 1: RETURN
     IF MD$ = "=" THEN ST = ST + 1
  WEND
  IF MD$ = "+" THEN ST = ST + 1
  IF MD$ = "-" THEN SYGN = -1: ST = ST + 1
'
' Find ending position EN of value
'
  EN = ST: VAAL = 0: MD$ = MID$(EQ$, EN, 1): L = LEN(EQ$)
  WHILE EN <= L AND (MD$ <> "X" AND MD$ <> "Y" AND MD$ <> "=")
    MD$ = MID$(EQ$, EN, 1)
    EN = EN + 1
  WEND
```

```
  EN = EN - 1
  IF MD$ = "X" OR MD$ = "Y" OR MD$ = "=" THEN EN = EN - 1
  IF MD$ = "=" THEN SYGN = -SYGN          'Bring C to other side
  IF ST > EN THEN   'No Value
    VAAL = SYGN: ST = ST + 1
  ELSE              'Determine Value
    MD$ = MID$(EQ$, ST, EN - ST + 1)
    VAAL = SYGN * VAL(MD$): ST = EN + 2
  END IF
  RETURN



'3.7
' This program displays all simi-perfect #s between 2 and 35.
'
DEFINT A-Z
DIM A(20), B(20)
PRINT "SEMI #  EXAMPLE(S)"
FOR NUM = 2 TO 34: MAX = 0
  FOR DIV = 1 TO NUM / 2
    IF NUM MOD DIV = 0 THEN MAX = MAX + 1: B(MAX) = DIV
  NEXT DIV
  FOR B = 2 TO MAX
    L = MAX: GOSUB Combo
  NEXT B
NEXT NUM: END
'
' Produce combinations
'
Combo:
  FOR I = 1 TO B: A(I) = B - I + 1: NEXT I
  A(1) = A(1) - 1: N = 1
'
  WHILE N <= B
    A(N) = A(N) + 1
    FOR I = N - 1 TO 1 STEP -1: A(I) = A(I + 1) + 1: NEXT I
    IF A(N) <= L - N + 1 THEN
      SUM = 0: FOR I = 1 TO B: SUM = SUM + B(A(I)): NEXT I
      IF SUM = NUM THEN
'
        PRINT USING "##"; NUM; : PRINT SPACE$(5); B(A(B));
        FOR I = B - 1 TO 1 STEP -1
          PRINT "+"; B(A(I));
        NEXT I: PRINT
      END IF
      N = 0
    END IF
    N = N + 1
  WEND
  RETURN
```

```
'3.8
' This program will keep score for a bowler.
'
DIM A(10, 3): CLS
INPUT "Enter frames:"; F$: F$ = F$ + " "
FOR I = 1 TO 10
  COMMAPOS = INSTR(F$, " ")
  A$(I) = MID$(F$, 1, COMMAPOS - 1)
  F$ = MID$(F$, COMMAPOS + 1, LEN(F$) - COMMAPOS)
NEXT I
PRINT
PRINT "-1- -2- -3- -4- -5- -6- -7- -8- -9- -10-"
PRINT "---!---!---!---!---!---!---!---!---!---!"
FOR I = 1 TO 10
  PRINT SPACE$(3 - LEN(A$(I))); A$(I); "!";
NEXT I
PRINT
'
' Assign values to A FRames according to X, /, or pins
'
FOR FR = 1 TO 10
  L = LEN(A$(FR))
  FOR J = 1 TO L
    MD$ = MID$(A$(FR), J, 1)
    IF MD$ = "X" THEN
      A(FR, J) = 10: LOOK(FR) = 2
    ELSE
      IF MD$ = "/" THEN
        A(FR, J) = 10 - A(FR, J - 1): LOOK(FR) = 1
      ELSE
        A(FR, J) = VAL(MD$)
      END IF
    END IF
  NEXT J
NEXT FR
'
' Determine FRame values with LOOK ahead
'
FOR FR = 1 TO 10
  SUM(FR) = SUM(FR - 1) + A(FR, 1) + A(FR, 2)
  IF LOOK(FR) > 0 THEN
    IF LOOK(FR) <= 1 THEN
'       ***  A spare / needs 1 more value added  ***
        IF FR = 10 THEN
          SUM(FR) = SUM(FR) + A(FR, 3)
        ELSE
          SUM(FR) = SUM(FR) + A(FR + 1, 1)
        END IF
    ELSE
 '       ***  A strike X needs 2 more values added  ***
      IF FR = 10 THEN
        SUM(FR) = SUM(FR) + A(FR, 3)
      ELSE
        SUM(FR) = SUM(FR) + A(FR + 1, 1) + A(FR + 1, 2)
        IF FR <> 9 THEN
```

```
          IF A(FR + 1, 1) = 10 THEN
            SUM(FR) = SUM(FR) + A(FR + 2, 1)
          END IF
        END IF
      END IF
    END IF
  END IF
 '      *** Print FRame's value ***
  SUM$ = MID$(STR$(SUM(FR)), 2)
  PRINT SUM$; SPACE$(3 - LEN(SUM$)); "!";
NEXT FR
PRINT : PRINT STRING$(40, "-")



'3.9
' This program will convert a real from one base to another.
'
INPUT "Enter M, N, #: "; M, N, NUM$
PRINT LEFT$(NUM$, 2);
NUM$ = MID$(NUM$, 3):
MDIGITS = LEN(NUM$)   'Digits on right of period(.)
'
NDIGITS = 1
WHILE (1 / N) ^ NDIGITS > (1 / M) ^ MDIGITS AND NDIGITS < 7
  NDIGITS = NDIGITS + 1
WEND
'
' SUM= Base 10 # of NUM$
'
FOR I = 1 TO MDIGITS
  MD$ = MID$(NUM$, I, 1)
  MD = ASC(MD$) - 48: IF MD > 9 THEN MD = MD - 7
  SUM = SUM + MD / (M ^ I)
NEXT I
'
' Convert base 10 decimal to Base N fraction
'
FOR I = 1 TO NDIGITS + 1
  SUM = SUM * N: NUM(I) = INT(SUM): SUM = SUM - NUM(I)
NEXT I
'
' Print fraction with last digit rounded according to NDIGIT+1
'
FOR I = 1 TO NDIGITS - 1
  PRINT CHR$(48 + NUM(I) - (NUM(I) > 9) * 7);
NEXT I
IF NUM(NDIGITS + 1) >= N / 2 THEN NUM(NDIGITS) = NUM(NDIGITS) + 1
PRINT CHR$(48 + NUM(NDIGITS) - (NUM(NDIGITS) > 9) * 7);
```

```
'3.10
' This program computes the composition of P(Q) and Q(P).
'
INPUT "Enter to the ORDER of p(x): "; PORDER
FOR I = PORDER TO 0 STEP -1
  PRINT "Enter coefficient for x**"; I; ": "; : INPUT PCO(I)
NEXT I: PRINT
INPUT "Enter to the ORDER of q(x): "; QORDER
FOR I = QORDER TO 0 STEP -1
  PRINT "Enter coefficient for x**"; I; ": "; : INPUT QCO(I)
NEXT I
PRINT "P(Q(X))= "; : GOSUB CompPofQ
PRINT
' *****  Swap P and Q to perform Q(P(X))  *****
SWAP PORDER, QORDER
IF PORDER > QORDER THEN MAX = PORDER ELSE MAX = QORDER
FOR I = 0 TO MAX: SWAP PCO(I), QCO(I): NEXT I
PRINT "Q(P(X))= "; : GOSUB CompPofQ
END
'
' *****  Compute composition P of Q  *****
'
CompPofQ:
  COMPORDER = PORDER * QORDER
  FOR I = 1 TO COMPORDER: POFQ(I) = 0: NEXT I
  FOR I = 0 TO PORDER
    IF PCO(I) <> 0 THEN
      IF I = 0 THEN
        POFQ(0) = PCO(0)
      ELSE
        FOR J = 0 TO QORDER: PROD(J) = QCO(J): NEXT J
        PRODORDER = QORDER
        IF I <> 1 THEN
          FOR IN = 1 TO I - 1
            FOR J = 0 TO PRODORDER: PROD2(J) = 0: NEXT J
            FOR J = 0 TO PRODORDER
              FOR K = 0 TO QORDER
                PROD2(J + K) = PROD2(J + K) + PROD(J) * QCO(K)
              NEXT K
            NEXT J
            PRODORDER = J + K
            FOR L = 0 TO PRODORDER
              PROD(L) = PROD2(L): PROD2(L) = 0
            NEXT L
          NEXT IN
        END IF
        FOR J = 0 TO PRODORDER
          PROD(J) = PROD(J) * PCO(I)
        NEXT J
        FOR J = PRODORDER TO 0 STEP -1
          POFQ(J) = POFQ(J) + PROD(J)
        NEXT J
      END IF
    END IF
  NEXT I
```

```
' *****  Print composition *****
  FOR I = COMPORDER TO 0 STEP -1
    IF I < COMPORDER THEN PRINT " + ";
    PRINT LTRIM$(STR$(POFQ(I))); "X**"; LTRIM$(STR$(I));
  NEXT I
RETURN
```