FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '96
BASIC PROGRAM SOLUTIONS

```
'1.1
' This program displays a phrase of the form FHSCC '##.
'
INPUT "Enter year:"; YEAR$
PRINT "FHSCC '"; MID$(YEAR$, 3, 2)
```

```
'1.2
' This program tallies number of frequent flier miles.
'
INPUT "Enter X:"; X
INPUT "Enter Y:"; Y
PRINT X * (1300 + 1300 + 500) + (Y * 5)
```

```
'1.3
' This program displays middle letter(s) of a word.
'
INPUT "Enter word:"; WORD$
L = LEN(WORD$): M = INT(L / 2)
IF L MOD 2 = 0 THEN PRINT MID$(WORD$, M, 1);
PRINT MID$(WORD$, M + 1, 1)
```

```
'1.4
' This program displays area and perimeter of a rectangle
'
INPUT "Enter coordinate 1:"; X1, Y1
INPUT "Enter coordinate 2:"; X2, Y2
AREA = ABS((X1 - X2) * (Y1 - Y2))
PERIM = (ABS(X1 - X2) + ABS(Y1 - Y2)) * 2
PRINT "AREA ="; AREA
PRINT "PERIMETER ="; PERIM
```

```
'1.5
' This program code-breaks an encrypted secret message.
'
INPUT "Enter encryption:"; E$
FOR I = 1 TO LEN(E$)
  M$ = MID$(E$, I, 1)
  IF M$ = " " THEN
    PRINT M$;
  ELSE
    PRINT CHR$(ASC("Z") - ASC(M$) + ASC("A"));
  END IF
NEXT I
PRINT
```

```
'1.6
' This program display number of floors touched by elevator
'
DO
  INPUT "Enter floor:"; FLOOR
  TOTAL = TOTAL + ABS(FLOOR - LASTFLOOR)
  IF FLOOR > MAX THEN MAX = FLOOR
  LASTFLOOR = FLOOR
LOOP UNTIL (FLOOR = 0)
' 1 is added for the starting ground floor
PRINT "TOTAL FLOORS TOUCHED ="; TOTAL + 1
PRINT "UNIQUE FLOORS TOUCHED ="; MAX + 1


'1.7
' This program displays a person's ratios for buying a house.
'
INPUT "Enter amount of loan:"; LOAN
INPUT "Enter amount of debts:"; DEBTS
INPUT "Enter amount of income:"; INCOME
RATIO1 = (LOAN / INCOME) * 100
RATIO2 = ((LOAN + DEBTS) / INCOME) * 100
PRINT USING "RATIOS = ##.#% / ##.#%"; RATIO1; RATIO2
PRINT "DOES ";
IF RATIO1 > 33 OR RATIO2 > 38 THEN PRINT "NOT ";
PRINT "QUALIFY"


'1.8
' This program will convert numbers to English or Spanish.
'
DATA ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE,TEN
DATA UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE,DIEZ
INPUT "Enter E or S:"; LANG$
INPUT "Enter number:"; NUM
IF LANG$ = "S" THEN FOR I = 1 TO 10: READ N$: NEXT I
FOR I = 1 TO NUM
  READ N$
NEXT I
PRINT N$


'1.9
' This program forms a cross from word(s).
'
INPUT "Enter word(s):"; W$
L = LEN(W$): M = INT(L / 2) + 1
FOR I = 1 TO L
  IF I <> M THEN
    PRINT SPACE$(M - 1); MID$(W$, I, 1)
  ELSE
    PRINT W$
  END IF
NEXT I
```

```
'1.10
' This program simulates the PRICE IS RIGHT game.
'
INPUT "Enter actual price:"; PRICE
INPUT "Enter guesses A, B, C, D"; A(1), A(2), A(3), A(4)
MIN = 32000
FOR I = 1 TO 4
  IF A(I) <= PRICE THEN
    DIF = PRICE - A(I)
    IF DIF < MIN THEN MIN = DIF: INDEX = I
  END IF
NEXT I
IF INDEX > 0 THEN
  PRINT "PERSON "; MID$("ABCD", INDEX, 1)
ELSE
  PRINT "EVERYONE IS OVER"
END IF
```

```
'2.1
' This program will emulate random dart throws.
'
DATA 0,2,4,5,10,20,50
FOR I = 1 TO 7: READ S$(I): NEXT I: PRINT " ";
RANDOMIZE TIMER
DO
  X = INT(RND(3) * 7) + 1: THROW = THROW + 1
  PRINT S$(X);
  TOTAL = TOTAL + VAL(S$(X))
  IF TOTAL < 100 THEN PRINT ",";
LOOP UNTIL TOTAL >= 100
PRINT : PRINT THROW; "THROWS ACHIEVED SCORE OF"; TOTAL: PRINT
```

```
'2.2
' This program compresses information to save space.
'
INPUT "Enter string:"; S$
FOR I = 1 TO LEN(S$)
  MD$ = MID$(S$, I, 1)
  IF MD$ <> "*" THEN
    IF AST > 0 THEN
      IF AST = 1 THEN PRINT "*";  ELSE PRINT USING "#"; AST;
      AST = 0
    END IF
    PRINT MD$;
  ELSE
    AST = AST + 1
  END IF
NEXT I
PRINT
```

```
'2.3
' This program finds 2 numbers to add to the set 1,3,8.
'
A(1) = 1: A(2) = 3: A(3) = 8: N = 3: I = 0
FOR I = 0 TO 999
  FOUND = -1
  FOR J = 1 TO N
    NUM = A(J) * I + 1
    IF SQR(NUM) - INT(SQR(NUM + .0001)) > .0001 THEN FOUND = 0
  NEXT J
  IF FOUND THEN
    PRINT I: N = N + 1: A(N) = I: IF N = 5 THEN END
  END IF
NEXT I
```

```
'2.4
' This program diplays the LCM of the first N integers.
'
DIM A(31): DEFDBL P
INPUT "Enter N:"; N
FOR I = 2 TO N: A(I) = I: NEXT I
' Produce all the necessary prime factors
FOR I = 2 TO N
  FOR J = I + 1 TO N
    IF A(J) MOD A(I) = 0 THEN A(J) = A(J) / A(I)
  NEXT J
NEXT I
'
PROD = 1
FOR I = 2 TO N: PROD = PROD * A(I): NEXT I
PRINT PROD
```

```
'2.5
' This program will calculate the fractional value.
'
INPUT "Enter word: "; A$
FOR I = 1 TO 3
  A(I) = ASC(MID$(A$, I, 1)) - 64
NEXT I
N = A(1) * A(2) + A(2) * A(3) + A(1) * A(3)
D = A(1) * A(2) * A(3)
FOR I = D TO 1 STEP -1
  IF N MOD I = 0 AND D MOD I = 0 THEN
    PRINT LTRIM$(STR$(N / I)); "/"; LTRIM$(STR$(D / I)): END
  END IF
NEXT I
```

```
'2.6
' This program displays the Nth prime in Fibonacci sequence.
'
DIM F(99)
F(1) = 1: F(2) = 1: F(3) = 2: PNUM = 1: I = 3
INPUT "Enter N:"; N
WHILE PNUM < N
  I = I + 1
  F(I) = F(I - 1) + F(I - 2): PRIME = -1
  ' Check if Fibonacci # is prime (not divisible by 2 or odd #)
  IF F(I) MOD 2 = 0 THEN PRIME = 0
  IF PRIME THEN
    FOR J = 3 TO SQR(F(I))
      IF F(I) MOD J = 0 THEN PRIME = 0
    NEXT J
    IF PRIME THEN PNUM = PNUM + 1
  END IF
WEND
PRINT F(I)
```

```
'2.7
' This program sorts phone bills by zip code and phone #.
'
DO
  N = N + 1
  INPUT "Enter phone #, zip:"; P$(N), Z$(N)
  PZ$(N) = Z$(N) + P$(N)
LOOP UNTIL (P$(N) = "0000") AND (Z$(N) = "00000")
N = N - 1
FOR I = 1 TO N - 1
  FOR J = I + 1 TO N
    IF PZ$(I) > PZ$(J) THEN
      SWAP PZ$(I), PZ$(J)
      SWAP P$(I), P$(J)
      SWAP Z$(I), Z$(J)
    END IF
  NEXT J
NEXT I
FOR I = 1 TO N: PRINT P$(I): NEXT I
```

```
'2.8
' This program will display number of runs of letters.
'
INPUT "Enter letters:"; LET$
FOR I = 1 TO LEN(LET$)
  CH$ = MID$(LET$, I, 1)
  IF INSTR("ABCDEFGHIJKLM", CH$) > 0 THEN
    IF HALF2 THEN H2 = H2 + 1: HALF2 = 0
    HALF1 = -1
  ELSE
    IF HALF1 THEN H1 = H1 + 1: HALF1 = 0
    HALF2 = -1
  END IF
NEXT I
IF HALF1 THEN H1 = H1 + 1
IF HALF2 THEN H2 = H2 + 1
PRINT "RUNS IN 1ST HALF ="; H1
PRINT "RUNS IN 2ND HALF ="; H2
```

```
'2.9
' This program reverses the order of letters in each word.
'
INPUT "Enter string:"; S$: S$ = S$ + " "
FOR I = 1 TO LEN(S$)
  MD$ = MID$(S$, I, 1)
  IF MD$ = " " THEN
    L = LEN(W$): PAL = -1
    FOR J = 1 TO L / 2
      IF MID$(W$, J, 1) <> MID$(W$, L - J + 1, 1) THEN PAL = 0
    NEXT J
    IF PAL THEN
      PRINT STRING$(LEN(W$), "?");
    ELSE
      FOR J = L TO 1 STEP -1: PRINT MID$(W$, J, 1); : NEXT J
    END IF
    PRINT " "; : W$ = ""
  ELSE
    W$ = W$ + MD$
  END IF
NEXT I
PRINT
```

```
'2.10
' This program determines day of week for a given date.
'
DIM MONNUM(12)
DATA 1,4,4,0,2,5,0,3,6,1,4,6
FOR I = 1 TO 12: READ MONNUM(I): NEXT I
INPUT "Enter month, day, year:"; MONTH, DAY, YEAR
LAST2 = YEAR MOD 100
SUM = LAST2 + INT(LAST2 / 4)
LEAPYEAR = (YEAR MOD 4 = 0) AND (YEAR MOD 100 > 0)
LEAPYEAR = LEAPYEAR OR (YEAR MOD 400 = 0)
IF (MONTH < 3) AND LEAPYEAR THEN
  IF MONTH = 2 THEN SUM = SUM + 3   'New Month Number
ELSE
  SUM = SUM + MONNUM(MONTH)
END IF
SUM = SUM + DAY
SELECT CASE YEAR
  CASE IS < 1800: SUM = SUM + 4
  CASE IS < 1900: SUM = SUM + 2
  CASE IS < 2000:
  CASE IS < 2100: SUM = SUM + 6
  CASE IS < 2200: SUM = SUM + 4
END SELECT
R = SUM MOD 7
DATA SATURDAY,SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY
FOR I = 1 TO R + 1: READ D$: NEXT I
PRINT D$
```

```
'3.1
' This program displays the appearance of 3-dimensional book.
'
INPUT "Enter title 1:"; T1$
INPUT "Enter title 2:"; T2$
IF LEN(T1$) > LEN(T2$) THEN
  MAX = LEN(T1$): DIF = INT((MAX - LEN(T2$)) / 2)
  T2$ = SPACE$(DIF) + T2$ + SPACE$(DIF + 1)
ELSE
  MAX = LEN(T2$): DIF = INT((MAX - LEN(T1$)) / 2)
  T1$ = SPACE$(DIF) + T1$ + SPACE$(DIF + 1)
END IF
CLS
PRINT "    /---/!"
PRINT "   /   / !"
PRINT "  /   /  !"
PRINT " /   /   !"
PRINT "!---!    !"
FOR ROW = 1 TO MAX
  PRINT "!";
  PRINT MID$(T2$, ROW, 1); " ";
  PRINT MID$(T1$, ROW, 1); "!";
  IF ROW < MAX - 3 THEN
    PRINT SPACE$(4); "!"
  ELSE
    PRINT SPACE$(MAX - ROW + 1); "/"
  END IF
NEXT ROW
PRINT "!---!/"
```

```
'3.2
' This program produces a prime factors tree.
'
DIM P(100)
INPUT "Enter number:"; NUM
CLS : PRINT TAB(5); NUM
LEFT = 5: RIGHT = LEFT + LEN(STR$(NUM))  'Position of / and \
DO
  ' Find smallest prime that divides number
  IF NUM MOD 2 = 0 THEN
    PR = 2
  ELSE
    PR = 1
    DO
      PR = PR + 2
    LOOP UNTIL (NUM MOD PR = 0)
  END IF
  DIVIDEND = NUM / PR
  IF DIVIDEND > 1 THEN
    PRINT TAB(LEFT); "/"; TAB(RIGHT); "\"
    LNUM$ = LTRIM$(STR$(PR)): RNUM$ = LTRIM$(STR$(DIVIDEND))
    L = LEN(LNUM$): R = LEN(RNUM$)
    PRINT TAB(LEFT - L); LNUM$; TAB(RIGHT + 1); RNUM$
    LEFT = RIGHT: RIGHT = RIGHT + R + 1
  END IF
  NUM = DIVIDEND
LOOP UNTIL NUM = 1
```

```
'3.3
' This program simulates a "base four" calculator.
'
INPUT "Enter base 4 expression:"; E$: E$ = E$ + "+"
SYM$(1) = "+"
FOR I = 1 TO LEN(E$)
  CH$ = MID$(E$, I, 1)
  IF CH$ = "+" OR CH$ = "-" THEN
    J = J + 1: NUM$(J) = N$: SYM$(J + 1) = CH$: N$ = ""
  ELSE
    N$ = N$ + CH$
  END IF
NEXT I
' Convert base 4 numbers to base 10 and perform arithmetic
FOR I = 1 TO J
  L = LEN(NUM$(I)): B10 = 0
  FOR J = 1 TO L
    DIG = VAL(MID$(NUM$(I), J, 1))
    B10 = B10 + DIG * 4 ^ (L - J)
  NEXT J
  IF SYM$(I) = "-" THEN B10 = (-B10)
  TOTAL = TOTAL + B10
NEXT I
' Convert base 10 number to base 4
IF TOTAL < 0 THEN PRINT "-"; : TOTAL = (-TOTAL)
```

```
J = INT(LOG(TOTAL) / LOG(4) + .001)
FOR I = J TO 0 STEP -1
  POW = 4 ^ I
  X = INT(TOTAL / POW): PRINT USING "#"; X;
  TOTAL = TOTAL - X * POW
NEXT I
PRINT


'3.4
' This program calculates contractor's pay = time * rate
'
INPUT "Enter pay/hour:"; RATE
INPUT "Enter start time:"; ST$
INPUT "Enter finish time:"; FI$
STHOUR = VAL(MID$(ST$, 1, 2))
FIHOUR = VAL(MID$(FI$, 1, 2))
STMIN = VAL(MID$(ST$, 4, 2))
FIMIN = VAL(MID$(FI$, 4, 2))
' Adjust for 12AM and times from 1PM - 11PM
IF STHOUR = 12 THEN
  IF MID$(ST$, 6, 2) = "AM" THEN STHOUR = STHOUR - 12
ELSE
  IF MID$(ST$, 6, 2) = "PM" THEN STHOUR = STHOUR + 12
END IF
IF FIHOUR = 12 THEN
  IF MID$(FI$, 6, 2) = "AM" THEN FIHOUR = FIHOUR - 12
ELSE
  IF MID$(FI$, 6, 2) = "PM" THEN FIHOUR = FIHOUR + 12
END IF
' Adjust for a late starting time and early morning finish
IF STHOUR > FIHOUR THEN FIHOUR = FIHOUR + 24
' Compute difference in time (finish - start)
TIME = (FIHOUR - STHOUR) + (FIMIN - STMIN) / 60
' If more than half of time is outside normal hours (7AM - 5PM)
' then add a shift differential of 10% to rate.
IF (7 - STHOUR) + (0 - STMIN) / 60 >= TIME / 2 THEN
  ' More than half of time is worked before 7AM
  RATE = RATE * 1.1
END IF
IF (FIHOUR - 17) + (FIMIN) / 60 >= TIME / 2 THEN
  ' More than half of time is worked after 5PM
  RATE = RATE * 1.1
END IF
PRINT USING "$###.##"; TIME * RATE
```

```
'3.5
' This program will display the button that leads to the others.
'
FOR I = 1 TO 4
  INPUT "Enter row:"; ROW$
  FOR J = 1 TO 4
    N(I, J) = VAL(MID$(ROW$, J * 3 - 2, 1))
    D$(I, J) = MID$(ROW$, J * 3 - 1, 1)
  NEXT J
NEXT I
FOR I = 1 TO 4
 FOR J = 1 TO 4
   FOR K = 1 TO 4: FOR L = 1 TO 4: A(K, L) = 0: NEXT L, K
   R = I: C = J: A(R, C) = -1: PRESS = 1: GOOD = -1
   DO
     SELECT CASE D$(R, C)
       CASE "D": R = R + N(R, C)
       CASE "U": R = R - N(R, C)
       CASE "L": C = C - N(R, C)
       CASE "R": C = C + N(R, C)
     END SELECT
     IF A(R, C) THEN
       GOOD = 0
     ELSE
       A(R, C) = -1: PRESS = PRESS + 1
     END IF
   LOOP UNTIL (NOT GOOD) OR (PRESS = 16)
   IF PRESS = 16 THEN
     PRINT USING "FIRST BUTTON = #"; N(I, J); : PRINT D$(I, J)
     PRINT "AT ROW = "; : PRINT USING "#"; I;
     PRINT USING ", COL = #"; J: END
   END IF
 NEXT J
NEXT I
```

```
'3.6
' This program will generate odd size magic squares.
'
INPUT "Enter order, first number, increment: "; N, FIRST, INC
DIM A(N, N)
X = 1: Y = (N + 1) / 2: A(X, Y) = FIRST
FOR I = 2 TO N * N
  X = X - 1: Y = Y + 1
  IF X = 0 THEN X = N
  IF Y > N THEN Y = 1
  IF A(X, Y) = 0 THEN
    A(X, Y) = FIRST + INC * (I - 1)
  ELSE
    X = X + 2: Y = Y - 1
    IF X > N THEN X = X - N
    IF Y = 0 THEN Y = N
    A(X, Y) = FIRST + INC * (I - 1)
  END IF
NEXT I
' Display Magic Number and Square
FOR I = 1 TO N: MAGICNUM = MAGICNUM + A(I, 1): NEXT I
PRINT "MAGIC NUMBER ="; MAGICNUM
FOR I = 1 TO N
  FOR J = 1 TO N
    PRINT USING "####"; A(I, J);
  NEXT J: PRINT
NEXT I
```

```
'3.7
' This program will generate 6x6 magic squares.
'
INPUT "Enter first number, increment: "; FIRSTN, INC
' Four 3x3 squares are made for the 6x6 matrix B()
' upper-left, bottom-right, upper-right, bottom-left
DATA 0,0, 1,1, 0,1, 1,0
FOR SQ = 0 TO 3
  FIRST = FIRSTN + SQ * 9 * INC
  GOSUB Generate3x3
  READ R, C
  FOR I = 1 TO 3
    FOR J = 1 TO 3
      B(R * 3 + I, C * 3 + J) = A(I, J)
    NEXT J
  NEXT I
NEXT SQ
' Transpose three cells
SWAP B(1, 1), B(4, 1)
SWAP B(2, 2), B(5, 2)
SWAP B(3, 1), B(6, 1)
' Display 6x6 matrix
FOR I = 1 TO 6: MAGICNUM = MAGICNUM + B(I, 1): NEXT I
PRINT "MAGIC NUMBER ="; MAGICNUM
FOR I = 1 TO 6
  FOR J = 1 TO 6
    PRINT USING "####"; B(I, J);
  NEXT J: PRINT
NEXT I
END

Generate3x3:  'Generate a 3x3 magic square in A(1..3,1..3)
  FOR I = 1 TO 3: FOR J = 1 TO 3: A(I, J) = 0: NEXT J, I
  N = 3
  X = 1: Y = (N + 1) / 2: A(X, Y) = FIRST
  FOR I = 2 TO N * N
    X = X - 1: Y = Y + 1
    IF X = 0 THEN X = N
    IF Y > N THEN Y = 1
    IF A(X, Y) = 0 THEN
      A(X, Y) = FIRST + INC * (I - 1)
    ELSE
      X = X + 2: Y = Y - 1
      IF X > N THEN X = X - N
      IF Y = 0 THEN Y = N
      A(X, Y) = FIRST + INC * (I - 1)
    END IF
  NEXT I
RETURN
```

```
'3.8
' This program will display a pie graph.
'
DIM A(21, 21)
INPUT "Enter 3 percentages: "; P(1), P(2), P(3)
A$(1) = "A": A$(2) = "D": A$(3) = "N"
CLS : PI = 3.14159
' Draw circle
FOR I = -PI / 2 TO 3 / 2 * PI STEP .1
  X = COS(I) * 10:  Y = SIN(I) * 10
  LOCATE 11 + Y, 11 + X: PRINT "*": A(11 + Y, 11 + X) = 1
NEXT I
' Draw 3 line segments from center
FOR S = 0 TO 2
  SUM = SUM + P(S)
  I = -PI / 2 + 2 * PI * SUM / 100
  FOR R = 0 TO 10
    X = COS(I) * R: Y = SIN(I) * R
    LOCATE 11 + Y, 11 + X: PRINT "*": A(11 + Y, 11 + X) = 1
  NEXT R
NEXT S
A$ = INPUT$(1):  SUM = 0
' Fill regions with letters
FOR S = 1 TO 3
  LSUM = SUM:  SUM = SUM + P(S)
  FOR L = LSUM TO SUM
    I = -PI / 2 + 2 * PI * L / 100
    FOR R = 1 TO 9
      X = COS(I) * R: Y = SIN(I) * R
      IF A(11 + Y, 11 + X) = 0 THEN
        LOCATE 11 + Y, 11 + X: PRINT A$(S)
      END IF
    NEXT R
  NEXT L
NEXT S
```

```
'3.9
' This program produces a precedence of jobs to run.
'
INPUT "Enter number of dependencies:"; NUM
FOR I = 1 TO NUM
  INPUT "Enter dependency:"; DEP$: DEP$ = DEP$ + " "
  A$(I) = MID$(DEP$, 1, 3)
  B$(I) = MID$(DEP$, 4, 3)
  ' Store unique jobs in string
  IF INSTR(U$, A$(I)) = 0 THEN U$ = U$ + A$(I)
  IF INSTR(U$, B$(I)) = 0 THEN U$ = U$ + B$(I)
NEXT I
' Since there is a unique order for all the jobs,
' every job will have its successor somewhere in B().
' 1) search all B() for the only job missing.
' 2) exclude all dependencies with this job in it.
' 3) search all B() for the next only job missing.
' 4) repeat steps 2 and 3 until the final dependency is left.
L = LEN(U$): UNUM = L / 3: U2$ = U$: DEPLEFT = NUM:  JOBS$ = ""
WHILE DEPLEFT > 1
  FOR I = 1 TO NUM: MARKED(I) = 0: NEXT I
  FOR I = 1 TO NUM
    P = INSTR(U2$, B$(I))
    IF P > 0 THEN MARKED((P + 2) / 3) = -1
  NEXT I
  NOJOB = -1: I = 0
  WHILE NOJOB AND (I < UNUM)
    I = I + 1: ST = I * 3 - 2
    JOB$ = MID$(U2$, ST, 3)
    VALIDJOB = (INSTR(JOBS$, JOB$) = 0) AND (JOB$ <> SPACE$(3))
    IF VALIDJOB AND NOT MARKED(I) THEN
      JOBS$ = JOBS$ + JOB$
      FOR K = 1 TO NUM
        IF A$(K) = JOB$ THEN
          A$(K) = "*": B$(K) = "*"
          DEPLEFT = DEPLEFT - 1
        END IF
      NEXT K
      NEWU2$ = MID$(U2$, 1, ST - 1) + SPACE$(3)
      U2$ = NEWU2$ + MID$(U2$, ST + 3, L - ST - 2)
      NOJOB = 0
    END IF
  WEND
WEND
' Last dependency is concatenated
FOR I = 1 TO NUM
  IF A$(I) <> "*" THEN JOBS$ = JOBS$ + A$(I) + B$(I)
NEXT I
PRINT "JOBS MUST BE RUN IN THIS ORDER: "; JOBS$
```

```
'3.10
' This program finds a perfect square with digits 1-9.
'
DEFINT B, Z: DEFLNG A, N: MIN = 9
FOR NUM = 10001 TO INT(SQR(987654321))
  A = NUM * NUM
  DIGITS$ = LTRIM$(STR$(A))
  GOOD = -1: L = 1
  WHILE (L <= 9) AND GOOD
    IF INSTR(DIGITS$, CHR$(48 + L)) = 0 THEN GOOD = 0
    L = L + 1
  WEND
  IF GOOD THEN         'Found perfect square with unique digits
    GOSUB CheckDigits  'Count will contain number of swaps made
    IF COUNT < MIN THEN MIN = COUNT: NUMMIN = A: NUMMIN2 = NUM
  END IF
NEXT NUM
' Display the perfect square needing least number of swaps
DIGITS$ = LTRIM$(STR$(NUMMIN))
PRINT DIGITS$; " IS THE SQUARE OF"; NUMMIN2
PRINT "AND WAS FORMED BY EXCHANGING"; MIN; "PAIRS OF DIGITS"
END

CheckDigits:  'Determine number of swaps made and store in count
  FOR I = 1 TO 9: A(I) = VAL(MID$(DIGITS$, I, 1)): NEXT I
  COUNT = 0
  FOR I = 1 TO 9
    IF A(I) <> I THEN
      J = I + 1
      WHILE J < 9 AND A(J) <> I
        J = J + 1
      WEND
      SWAP A(I), A(J): COUNT = COUNT + 1
    END IF
  NEXT I
  RETURN
```