FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
BASIC PROGRAM SOLUTIONS

```
'1.1
' This program displays title of contest forward and backward.
'
A$ = "FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95"
FOR I = 1 TO 4
    PRINT A$
    FOR J = LEN(A$) TO 1 STEP -1
      PRINT MID$(A$, J, 1);
    NEXT J
    PRINT
NEXT I


'1.2
' This program will generate comments in different languages.
'
INPUT "Enter comment:"; C$
PRINT "BASIC: ' "; C$
PRINT "PASCAL: { "; C$; " }"
PRINT "C: /* "; C$; " */"
PRINT "C++: // "; C$


'1.3
' This program either increments or decrements N by 1.
'
INPUT "Enter N:"; N
INPUT "Enter operator:"; OP$
IF OP$ = "++" THEN
  PRINT N + 1
ELSE          'OPerator is "--"
  PRINT N - 1
END IF


'1.4
' This program rounds to three decimal places by break point
'
INPUT "Enter break point:"; BP
INPUT "Enter number:"; NUM
ROUND = INT(NUM * 1000 + (10 - BP) / 10) / 1000
PRINT USING "#.###"; ROUND
```

```
'1.5
' This program will determine if a program is a REXX or a CLIST.
'
INPUT "Enter comment:"; C$
IF INSTR(C$, "REXX") > 0 THEN
  PRINT "REXX"
ELSE
  PRINT "CLIST"
END IF


'1.6
' This program displays the number of times variables appear.
'
INPUT "Enter number of variables:"; NUM
INPUT "Enter number initialized:"; INIT
INPUT "Enter number initialized to 0:"; INIT0
PRINT "BASIC ="; INIT - INIT0
PRINT "PASCAL ="; NUM + INIT
PRINT "C/C++ ="; NUM


'1.7
' This program displays the last qualifier of a data set name.
'
INPUT "Enter data set name"; DSN$
FOR I = LEN(DSN$) TO 1 STEP -1
  CH$ = MID$(DSN$, I, 1)
  IF CH$ = "." THEN
    PRINT LAST$: END
  ELSE
    LAST$ = CH$ + LAST$
  END IF
NEXT I


'1.8
' This program displays a set of real numbers in reverse order.
'
INPUT "Enter N:"; N
FOR I = 1 TO N
  INPUT "Enter #:"; A$(I)
NEXT I
PRINT
FOR I = N TO 1 STEP -1
  PRINT A$(I)
NEXT I
```

```
'1.9
' This program displays a large X made up of letter X's.
'
INPUT "Enter number of X's:"; NUM
CLS
FOR I = 1 TO NUM
  LOCATE I, I: PRINT "X"
  LOCATE I, NUM - I + 1: PRINT "X"
NEXT I
```

```
'1.10
' This program will display the savings in postage.
'
COST = 23.33333
INPUT "Enter # of printed sides:"; PS
INPUT "Enter # of single sided pages:"; SS
' Calculate # of pages and weight for 1st bill
PAGE1 = PS - 6: OZ1 = 1
OZ1 = OZ1 + INT((PAGE1 + 8) / 9)
' Calculate # of pages and weight for 2nd bill
PAGE2 = SS + INT((PS - SS + 1) / 2) - 6
OZ2 = 1
OZ2 = OZ2 + INT((PAGE2 + 8) / 9)
PRINT USING "###.## CENTS SAVED"; (OZ1 - OZ2) * COST
```

```
'2.1
' This program finds integral solutions of (X,Y) for AX + BY = C.
'
INPUT "Enter A, B, C:"; A, B, C
X = 1
DO
  Y = (C - A * X) / B
  IF ABS(Y - INT(Y)) < .001 THEN
    PRINT "("; LTRIM$(STR$(X)); ","; LTRIM$(STR$(Y)); ")"
    END
  END IF
  X = X + 1
LOOP UNTIL X > 10000


'2.2
' This program verifies a part number by validating check digit
'
INPUT "Enter part number:"; PART$
L = LEN(PART$): PROD = 1
FOR I = 1 TO L - 1
  DIGIT = VAL(MID$(PART$, I, 1))
  SUM = SUM + DIGIT * ((I MOD 2) + 1)
NEXT I
' Subtract units digit of sum from 9 for check digit
CHKDIGIT = 9 - (SUM MOD 10)
IF CHKDIGIT = VAL(RIGHT$(PART$, 1)) THEN
  PRINT "OKAY"
ELSE
  PRINT "ERROR - CHECK DIGIT SHOULD BE"; CHKDIGIT
END IF


'2.3
' This program determines number of prizes given of $13 million
'
PRIZE = 13000000
' Same algorithm is used as converting # to base 13 #
FOR I = 6 TO 0 STEP -1
  POW(I) = INT(13 ^ I + .1)
  A(I) = INT(PRIZE / POW(I))
  PRIZE = PRIZE MOD POW(I)
NEXT I
FOR I = 0 TO 6
  PRINT "$"; LTRIM$(STR$(POW(I))); " ="; A(I)
NEXT I
```

```
'2.4
' This program will determine the cost of Directory Assistance.
'
INPUT "Enter number of DACs:"; N
FOR I = 1 TO N
  INPUT "Enter DAC:"; DAC$
  IF DAC$ = "00" THEN
    COST = 3!
  ELSEIF DAC$ = "1411" THEN
    LOCALDAC = LOCALDAC + 1: COST = 0
  ELSE
    AREA$ = MID$(DAC$, 2, 3)
    IF AREA$ = "813" THEN
      COST = .25
    ELSEIF AREA$ = "305" OR AREA$ = "407" OR AREA$ = "904" THEN
      COST = .4
    ELSE
      COST = .65
    END IF
  END IF
  TOT = TOT + COST
NEXT I
' Every local DAC after the third cost 25 cents
IF LOCALDAC > 3 THEN
  TOT = TOT + (LOCALDAC - 3) * .25
END IF
PRINT USING "##.## DOLLARS"; TOT


'2.5
' This program will display the heading of even/odd pages.
'
DATA PROBLEMS,180,JUDGING CRITERIA,140
DATA BASIC SOLUTIONS,200,PASCAL SOLUTIONS,260
FOR I = 1 TO 4: READ P$(I), PNUM(I): NEXT I
'
INPUT "Enter page number:"; PAGE
IF PAGE MOD 2 = 0 THEN
  PRINT LTRIM$(STR$(PAGE));
  PRINT "  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION";
  PRINT " 1985 - 1994"
ELSE
  PRINT "FHSCC '";
  I = 1: PAG = PAGE
  WHILE PAG > PNUM(I)
    PAG = PAG - PNUM(I): I = I + 1
  WEND
  CH = INT(PAG / (PNUM(I) / 10))
  PRINT USING "## "; 85 + CH;
  PRINT P$(I); " ";
  PRINT PAGE
END IF
```

```
'2.6
' This program will compute the total ESTIMATED PREPARATION TIME.
'
DATA 1040,A,B,C,D,E
DATA 3,8,  2,53, 4,41, 0,53
DATA 2,32, 0,26, 1,10, 0,27
DATA 0,33, 0,8,  0,17, 0,20
DATA 6,26, 1,10, 2,5,  0,35
DATA 0,51, 0,42, 1,1,  0,41
DATA 2,52, 1,7, 1,16,  0,35
FOR I = 1 TO 6: READ FORM$(I): NEXT I
FOR I = 1 TO 6
  FOR J = 1 TO 4
    READ HR(I, J), MIN(I, J)
  NEXT J
NEXT I
' Tally form times until invalid entry
I = 0
DO UNTIL I > 6
  INPUT "Enter form:"; F$
  I = 1
  WHILE (I < 7) AND (F$ <> FORM$(I)): I = I + 1: WEND
  IF I < 7 THEN
    FOR J = 1 TO 4
      TOTHR = TOTHR + HR(I, J)
      TOTMIN = TOTMIN + MIN(I, J)
    NEXT J
  END IF
LOOP
'
TOTHR = TOTHR + INT(TOTMIN / 60)
TOTMIN = TOTMIN MOD 60
PRINT TOTHR; "HR.,"; TOTMIN; "MIN."
```

```
'2.7
' This program will calculate investments at GTE.
'
BEGPRICE = 32! * .85
RETURN401K = .14
'
INPUT "Enter salary:"; SALARY
INPUT "Enter 401K %:"; PERCENT: PERCENT = PERCENT / 100
MAXSHARES = INT(SALARY / 100)
PRINT "YOU CAN PURCHASE UP TO"; MAXSHARES; "SHARES"
INPUT "Enter number of shares:"; SHARES
INPUT "Enter end of year price:"; ENDPRICE
'
EMPCONT = SALARY * PERCENT
IF PERCENT >= .06 THEN
  COMPCONT = (SALARY * .06) * .75
ELSE
  COMPCONT = (SALARY * PERCENT) * .75
END IF
K401 = (EMPCONT + COMPCONT) * RETURN401K
STOCKGAIN = SHARES * (ENDPRICE - BEGPRICE)
TOTALGAIN = COMPCONT + K401 + STOCKGAIN
'
PRINT USING "COMPANY CONTRIBUTION: #####.##"; COMPCONT
PRINT USING "401K INTEREST RETURN: #####.##"; K401
PRINT USING "         STOCK GAINS: #####.##"; STOCKGAIN
PRINT USING "         TOTAL GAINS: #####.##"; TOTALGAIN
```

```
'2.8
' This program will produce loops of a spiral using letters.
'
INPUT "Enter number of spiral loops:"; NUM
INPUT "Enter first letter:"; LET$
CLS
ROW = 12: COL = 40: INCR = 1
WHILE LOOPNUM < NUM
  INCR = INCR + 2
  ' Go right
  LOCATE ROW, COL: PRINT STRING$(INCR, LET$)
  COL = COL + INCR - 1
  ' Go down
  FOR I = 1 TO INCR - 1
    LOCATE ROW + I, COL: PRINT LET$
  NEXT I
  ROW = ROW + INCR - 1: INCR = INCR + 2
  ' Go left
  COL = COL - INCR + 1
  LOCATE ROW, COL: PRINT STRING$(INCR, LET$)
  ' Go up
  FOR I = 1 TO INCR - 2
    LOCATE ROW - I, COL: PRINT LET$
  NEXT I
  ROW = ROW - INCR + 1
  IF LET$ = "Z" THEN LET$ = "A" ELSE LET$ = CHR$(ASC(LET$) + 1)
  LOOPNUM = LOOPNUM + 1
WEND
```

```
'2.9
' This program shows all possible moves for a Queen in chess.
'
INPUT "Enter column and row:"; RC$
COL = ASC(LEFT$(RC$, 1)) - ASC("A") + 1
ROW = 9 - VAL(RIGHT$(RC$, 1))
CLS
FOR I = 8 TO 1 STEP -1: PRINT USING "#"; I: NEXT I
PRINT "  A B C D E F G H"
' Horizontal moves
LOCATE ROW, 3: PRINT "* * * * * * * *"
' Vertical moves
FOR I = 1 TO 8: LOCATE I, COL * 2 + 1: PRINT "*": NEXT I
' Diagonal moves
FOR I = 1 TO 7
  R(1) = ROW - I: C(1) = COL - I
  R(2) = ROW + I: C(2) = COL + I
  R(3) = ROW - I: C(3) = COL + I
  R(4) = ROW + I: C(4) = COL - I
  FOR J = 1 TO 4
    IF R(J) > 0 AND R(J) < 9 AND C(J) > 0 AND C(J) < 9 THEN
       LOCATE R(J), C(J) * 2 + 1: PRINT "*"
    END IF
  NEXT J
NEXT I
LOCATE ROW, COL * 2 + 1: PRINT "Q"
```

```
'2.10
' This program tabulates information during a pre-election.
'
DATA MALE,FEMALE,50 AND BELOW,OVER 50,WHITE,OTHERS
DATA ABOVE $25000,$25000 AND BELOW
DATA WHITE MALE OVER 50 AND ABOVE $25000,OTHER
INPUT "Enter sex:"; SEX$
WHILE SEX$ <> "E"
  INPUT "Enter age:"; AGE
  INPUT "Enter race:"; RACE$
  INPUT "Enter income:"; INCOME
  INPUT "Enter party:"; PARTY$
  IF PARTY$ = "D" THEN COL = 1 ELSE COL = 2
  IF SEX$ = "M" THEN ROW = 1 ELSE ROW = 2
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF AGE <= 50 THEN ROW = 3 ELSE ROW = 4
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF RACE$ = "W" THEN ROW = 5 ELSE ROW = 6
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF INCOME > 25000 THEN ROW = 7 ELSE ROW = 8
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF RACE$ = "W" AND SEX$ = "M" AND AGE > 50 AND ROW = 7 THEN
    ROW = 9
  ELSE
    ROW = 10
  END IF
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  TOTAL = TOTAL + 1: PRINT
  INPUT "Enter sex:"; SEX$
WEND
'
PRINT TAB(33); "DEMOCRATIC  REPUBLICAN";
FOR ROW = 1 TO 10
  IF ROW MOD 2 = 1 THEN PRINT
  READ A$(ROW): PRINT A$(ROW);
  PRINT TAB(38);
  PRINT USING "###.#"; SUM(ROW, 1) / TOTAL * 100;
  PRINT USING "       ###.#"; SUM(ROW, 2) / TOTAL * 100
NEXT ROW
```

```
'3.1
' This program will determine how much IRS owes/pays.
'
DATA 22750, 55100, 115000, 250000, 9999999
FOR I = 1 TO 5: READ AMOUNT(I): NEXT I
DATA .15, .28, .31, .36, .396
FOR I = 1 TO 5: READ RATE(I): NEXT I
STDEDUCT = 3800: EXEMPTION = 2450
'
INPUT "Enter adjusted gross income:"; GROSS
INPUT "Enter itemized deductions:"; DEDUCTIONS
INPUT "Enter federal income tax withheld"; FEDTAX
IF DEDUCTIONS > STDEDUCT THEN
  INCOME = GROSS - DEDUCTIONS
ELSE
  INCOME = GROSS - STDEDUCT
END IF
TAXINC = INCOME - EXEMPTION
'
FOR I = 1 TO 5
  IF TAXINC <= AMOUNT(I) THEN
    FOR J = 1 TO I - 1
      TAX = TAX + (AMOUNT(J) - AMOUNT(J - 1)) * RATE(J)
    NEXT J
    TAX = TAX + (TAXINC - AMOUNT(I - 1)) * RATE(I)
    PRINT USING "######.## DOLLARS "; ABS(TAX - FEDTAX);
    IF FEDTAX < TAX THEN
      PRINT "YOU OWE"
    ELSE
      PRINT "WILL BE REFUNDED TO YOU"
    END IF: END
  END IF
NEXT I
```

```
'3.2
' This program will display a simplified phone bill.
'
L = 1: INPUT "Enter MIN:"; MIN(L)
WHILE MIN(L) > 0
  INPUT "Enter time:"; TIM$(L)
  L = L + 1
  INPUT "Enter MIN:"; MIN(L)
WEND
L = L - 1
'
' Display bill
PRINT "  BOB SMITH  (813) 555-1234": PRINT
PRINT "  TIME OF DAY  MIN.  CHARGE"
FOR I = 1 TO L
  IF LEFT$(TIM$(I), 1) = "0" THEN
    PRINT " "; MID$(TIM$(I), 2);
  ELSE
    PRINT TIM$(I);
  END IF
  ' Calculate charge
  HH = VAL(LEFT$(TIM$(I), 2))
  AM$ = MID$(TIM$(I), 7, 2)
  DAY$ = RIGHT$(TIM$(I), 3)
  BOOL1 = (HH > 7 AND HH < 12 AND AM$ = "AM")
  BOOL2 = (HH = 12 AND AM$ = "PM") OR (HH < 5 AND AM$ = "PM")
  MIDDAY = BOOL1 OR BOOL2
  IF HH > 4 AND HH < 11 AND AM$ = "PM" AND DAY$ <> "SAT" THEN
    RATE1 = .21: RATE2 = .16
  ELSEIF MIDDAY AND DAY$ <> "SAT" AND DAY$ <> "SUN" THEN
    RATE1 = .28: RATE2 = .21
  ELSE
    RATE1 = .14: RATE2 = .11
  END IF
  CHARGE(I) = RATE1 + RATE2 * (MIN(I) - 1)
  PRINT USING "  ###"; MIN(I);
  PRINT USING "   ###.##"; CHARGE(I)
  TOT = TOT + CHARGE(I)
NEXT I
IF TOT > 20 THEN DISC = TOT * .2
PRINT
PRINT "TOTAL CHARGES"; TAB(22);
PRINT USING "###.##"; TOT
PRINT "DISCOUNT"; TAB(22);
PRINT USING "###.##"; DISC
PRINT "CHARGES - DISCOUNT"; TAB(22);
PRINT USING "###.##"; TOT - DISC
```

```
'3.3
' This program simulates a baseball game.
'
DEFINT A-W
RANDOMIZE TIMER
CLS : PRINT
PRINT SPACE$(8);
FOR I = 1 TO 9: PRINT I; : NEXT I: PRINT " SCORE"
PRINT SPACE$(8); : FOR I = 1 TO 33: PRINT "-"; : NEXT I: PRINT
PRINT "TEAM A !"; SPACE$(27); "!"
PRINT "TEAM B !"; SPACE$(27); "!"
FOR IN = 1 TO 9
  FOR T = 1 TO 2
    S = 0: B = 0: W = 0: R = 0: O = 0
    WHILE O < 3
      X = RND(3)
      IF X < .4 THEN S = S + 1: STOT = STOT + 1
      IF X >= .4 THEN B = B + 1: BTOT = BTOT + 1
      IF S = 3 THEN O = O + 1: OTOT = OTOT + 1: S = 0: W = 0
      IF B = 4 THEN W = W + 1: WTOT = WTOT + 1: B = 0: S = 0
      IF W = 4 THEN R = R + 1: R(T) = R(T) + 1: W = 3
    WEND
    LOCATE 3 + T, 6 + IN * 3: PRINT R;
  NEXT T
NEXT IN
LOCATE 4, 39: PRINT USING "##"; R(1)
LOCATE 5, 39: PRINT USING "##"; R(2)
PRINT
PRINT "TOTAL # OF STRIKES:"; STOT
PRINT "TOTAL # OF BALLS:"; BTOT
PRINT "TOTAL # OF WALKS:"; WTOT
PRINT "TOTAL # OF STRIKE OUTS:"; OTOT
```

```
'3.4
' This program will produce all possible subsets of letters.
'
DEFINT A-Z: DIM SUB$(1024)
INPUT "Enter letters:"; L$
L = LEN(L$)
FOR I = 1 TO L: A$(I) = MID$(L$, I, 1): NEXT I
'
' Sort letters in A$()
FOR I = 1 TO L - 1
  FOR J = I + 1 TO L
    IF A$(I) > A$(J) THEN SWAP A$(I), A$(J)
  NEXT J
NEXT I
'
' Generate binary numbers to produce all subsets.
FOR N = 0 TO 2 ^ L - 1
  NUM = N
  FOR J = L - 1 TO 0 STEP -1
    BIT = INT(NUM / 2 ^ J)
    IF BIT THEN
      SUB$(N) = SUB$(N) + A$(L - J): NUM = NUM - 2 ^ J
    END IF
  NEXT J
NEXT N
'
' Bubble Sort subsets
FOR I = 0 TO 2 ^ L - 2
  FOR J = I + 1 TO 2 ^ L - 1
    IF SUB$(I) > SUB$(J) THEN SWAP SUB$(I), SUB$(J)
  NEXT J
NEXT I
'
' Display subsets
FOR I = 0 TO 2 ^ L - 1
  SUBLEN = LEN(SUB$(I)) + 3
  IF COL + SUBLEN > 50 THEN PRINT : COL = 0
  PRINT "{"; SUB$(I); "} ";
  COL = COL + SUBLEN
NEXT I
PRINT : PRINT "TOTAL SUBSETS ="; 2 ^ L
```

```
'3.5
' This program will sum big integers from 1 to N.
' Gauss's formula: SUM = N * (N+1) / 2.
'
DIM A(80), B(80), PROD(80), D(80)
INPUT "Enter N:"; N$
'
' Store digits of N$ in A() and B()
LENA = LEN(N$): LENB = LENA
FOR I = 1 TO LENA
  A(I) = VAL(MID$(N$, LENA - I + 1, 1))
  B(I) = A(I)
NEXT I
'
' Add 1 to number in B()
B(1) = B(1) + 1: I = 1
WHILE B(I) = 10
  B(I) = 0: I = I + 1: B(I) = B(I) + 1
WEND
IF I > LENB THEN LENB = I
'
' Multiply A() by B()
FOR I = 1 TO LENA
  CARRY = 0
  FOR J = 1 TO LENB
    S = I + J - 1
    PROD(S) = PROD(S) + A(I) * B(J) + CARRY
    CARRY = INT(PROD(S) / 10)
    PROD(S) = PROD(S) - CARRY * 10
  NEXT J
  IF CARRY > 0 THEN PROD(S + 1) = CARRY
NEXT I
IF CARRY > 0 THEN S = S + 1
'
' Divide product PROD() by 2
IF PROD(S) = 1 THEN S = S - 1: CARRY = 10
FOR I = S TO 1 STEP -1
  D(I) = INT((PROD(I) + CARRY) / 2)
  CARRY = (PROD(I) MOD 2) * 10
NEXT I
'
' Display answer in D()
FOR I = S TO 1 STEP -1
  PRINT USING "#"; D(I);
NEXT I: PRINT
```

```
'3.6
' This program will assign values to variables in BASIC code.
'
DO
  L = L + 1
  INPUT "Enter line:"; A$(L)
LOOP UNTIL A$(L) = "END"
L = L - 1
'
FOR I = 1 TO L
  ' Determine if first variable is new or old
  V$ = LEFT$(A$(I), 1)
  POSV = INSTR(ALLV$, V$)
  IF POSV = 0 THEN
    ALLV$ = ALLV$ + V$
    POSV = LEN(ALLV$)
  END IF
  '
  ' Assign value for first number
  CH$ = MID$(A$(I), 3, 1)
  IF CH$ >= "0" AND CH$ <= "9" THEN
    NUM1 = VAL(CH$)
  ELSE
    POSV2 = INSTR(ALLV$, CH$)
    NUM1 = B(POSV2)
  END IF
  '
  IF LEN(A$(I)) = 3 THEN
    ' Assign first number to current variable
    B(POSV) = NUM1
  ELSE
    ' Assign value for second number
    CH$ = RIGHT$(A$(I), 1)
    IF CH$ >= "0" AND CH$ <= "9" THEN
      NUM2 = VAL(CH$)
    ELSE
      POSV3 = INSTR(ALLV$, CH$)
      NUM2 = B(POSV3)
    END IF
    ' Perform operation with 1st and 2nd num and place in var
    OP$ = MID$(A$(I), 4, 1)
    SELECT CASE OP$
      CASE "+": B(POSV) = NUM1 + NUM2
      CASE "-": B(POSV) = NUM1 - NUM2
      CASE "*": B(POSV) = NUM1 * NUM2
      CASE "/": B(POSV) = NUM1 / NUM2
    END SELECT
  END IF
NEXT I
' Display the variables in order of appearance with values
FOR I = 1 TO LEN(ALLV$)
  PRINT MID$(ALLV$, I, 1); "=";
  PRINT LTRIM$(STR$(B(I)))
NEXT I
```

```
'3.7
' This program finds three 3-digit primes having digits 1-9.
'
' Generate primes into A()
DEFINT B-Z: DEFLNG A: DIM A(200)
FOR I = 101 TO 997 STEP 2
  J = 3: PRIME = -1
  WHILE (J <= SQR(I)) AND PRIME
    IF I MOD J = 0 THEN PRIME = 0
    J = J + 2
  WEND
  IF PRIME THEN  'Ensure that Digits are unique and not 0
    H = INT(I / 100)
    T = INT((I - H * 100) / 10)
    ONE = I - H * 100 - T * 10
    IF T > 0 AND H <> T AND T <> ONE AND H <> ONE THEN
      P = P + 1: A(P) = I
    END IF
  END IF
NEXT I
' Add the different combinations of 3 primes
FOR I = 1 TO P - 2
  FOR J = I + 1 TO P - 1
    FOR K = J + 1 TO P
      SUM = A(I) + A(J) + A(K)
      ' Check if SUM has 4 digits in ascending order
      IF SUM >= 1234 THEN
        DIGITS$ = LTRIM$(STR$(SUM)): GOOD = -1: L = 1
        DO
          IF MID$(DIGITS$, L, 1) >= MID$(DIGITS$, L + 1, 1) THEN
            GOOD = 0
          END IF
          L = L + 1
        LOOP UNTIL (L = 4) OR NOT GOOD
        ' Check all 3-digit primes for digits 1 through 9
        IF GOOD THEN
          ADIGITS = (A(I) * 1000 + A(J)) * 1000 + A(K)
          DIGITS$ = LTRIM$(STR$((ADIGITS))): L = 1
          WHILE (L <= 9) AND GOOD
            IF INSTR(DIGITS$, CHR$(48 + L)) = 0 THEN GOOD = 0
            L = L + 1
          WEND
          IF GOOD THEN
            PRINT A(I); "+"; A(J); "+"; A(K); "="; SUM
            PNUM = PNUM + 1: IF PNUM = 7 THEN END
          END IF
        END IF
      END IF
    NEXT K
  NEXT J
NEXT I
```

```
'3.8
' This program will display time MM:SS in block letters.
'
DATA ****     *  ****  ****  *  *  ****  *     ****  ****  ****
DATA *  *     *     *     *  *  *  *     *        *  *  *  *  *
DATA *  *     *  ****  ****  ****  ****  ****     *  ****  ****
DATA *  *     *  *        *     *     *  *  *     *  *  *     *
DATA ****     *  ****  ****     *  ****  ****     *  ****     *
DATA 6,10,6,10, 1,7,18,24
FOR I = 1 TO 5
  READ B$
  FOR J = 0 TO 9: A$(I, J) = MID$(B$, J * 6 + 1, 4): NEXT J
NEXT I
FOR I = 1 TO 4: READ MAX(I): NEXT I  'Maximum units for MM:SS
FOR I = 1 TO 4: READ COL(I): NEXT I  'Columns to start blocks
'
INPUT "Enter MM:SS:"; MMSS$
FOR I = 1 TO 4
  DIG(I) = VAL(MID$(MMSS$, I - (I > 2), 1))
NEXT I
'
CLS
LOCATE 2, 14: PRINT "*": LOCATE 4, 14: PRINT "*"
DO UNTIL CH$ <> ""
  FOR I = 1 TO 4
    FOR J = 1 TO 5
      LOCATE J, COL(I): PRINT A$(J, DIG(I))
    NEXT J
  NEXT I
  DIG(4) = DIG(4) + 1
  FOR J = 4 TO 1 STEP -1
    IF DIG(J) = MAX(J) THEN
      DIG(J - 1) = DIG(J - 1) + 1: DIG(J) = 0
    END IF
  NEXT J
  FOR I = 1 TO 3000: NEXT I  'Approximately 1 second
  CH$ = INKEY$
LOOP
```

```
'3.9
' This program will calculate the area of a polygon room.
'
INPUT "Enter number of sides:"; SIDES
FOR I = 1 TO SIDES
  INPUT "Enter movement:"; MOV$
  DIR$(I) = MID$(MOV$, 1, 1)
  L = LEN(MOV$)
  MOV$ = MID$(MOV$, 2, L - 1)
  DIST(I) = VAL(MOV$)
' Subtract Down and Left directions
  IF DIR$(I) = "D" OR DIR$(I) = "L" THEN DIST(I) = -DIST(I)
NEXT I
' Multiply length by width to obtain rectangle area,
' then add or subtract area from overall area.
I = 1: SUM = 0: AREA = 0
WHILE (I <= SIDES)
  SUM = SUM + DIST(I)
  AREA = AREA + (SUM * DIST(I + 1))
  I = I + 2
WEND
PRINT "AREA ="; ABS(AREA); "SQUARE FEET"



'3.10
' This program displays versions of libraries on a graph.
'
INPUT "Enter version #:"; Vers
INPUT "Enter first week in test:"; FirstWk
INPUT "Enter first week to display, # of weeks:"; FWKDisp, WkNum
CLS
LWKDisp = FWKDisp + WkNum - 1
' Display week #s at top (units first, then tens)
PRINT SPACE$(9);
FOR I = FWKDisp TO LWKDisp
  PRINT USING "#"; INT(I / 10);
NEXT I
PRINT : PRINT SPACE$(9);
FOR I = FWKDisp TO LWKDisp
  PRINT USING "#"; I MOD 10;
NEXT I
PRINT : PRINT
LastWk = FirstWk + 17
' Compute # of versions to backup from Vers input
Backup = INT((LastWk - FWKDisp) / 6)
Vers = Vers - Backup
FirstWk = FirstWk - 6 * Backup: LastWk = LastWk - 6 * Backup
DO UNTIL FirstWk > LWKDisp
  ' Display Version and indent
  PRINT "R1V"; RIGHT$(STR$(100 + Vers), 2); "L01 ";
  IF FWKDisp <= FirstWk THEN
    Min = FirstWk
    PRINT SPACE$(FirstWk - FWKDisp);
  ELSE
```

```
    Min = FWKDisp
  END IF
  IF LWKDisp >= LastWk THEN Max = LastWk ELSE Max = LWKDisp
  ' Display TestArea of 1 if Vers even, 2 if odd; P = Production
  TestArea = (Vers MOD 2) + 1
  FOR I = Min TO Max
    IF I < FirstWk + 12 THEN
      PRINT USING "#"; TestArea;
    ELSE
      PRINT "P";
    END IF
  NEXT I
  PRINT
  ' Display Pre-Production Version
  FirstPreWk = FirstWk + 5: LastPreWk = FirstWk + 10
  IF (LastPreWk >= FWKDisp) AND (FirstPreWk <= LWKDisp) THEN
    PRINT "R1V"; RIGHT$(STR$(100 + Vers - 1), 2); "L88 ";
    IF FirstPreWk > FWKDisp THEN
      Min = FirstPreWk
      PRINT SPACE$(FirstPreWk - FWKDisp);
    ELSE
      Min = FWKDisp
    END IF
    IF LWKDisp >= LastPreWk THEN
      Max = LastPreWk
    ELSE
      Max = LWKDisp
    END IF
    PRINT STRING$(Max - Min + 1, "*")
  END IF
  FirstWk = FirstWk + 6: LastWk = LastWk + 6
  Vers = Vers + 1
LOOP
```