FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '89
BASIC PROGRAM SOLUTIONS

```
'1.1
' This program will print an indented phrase on each line.
'
CLS : P$ = "1989 COMPUTER CONTEST"
FOR I = 1 TO 22:  PRINT SPACE$(I); P$:  NEXT I


'1.2
' This program will translate gigabytes to megabytes.
'
INPUT "Enter number of gigabytes:"; G
PRINT G * 1024; "MEGABYTES"


'1.3
' This program displays a word in a backward-L format.
'
INPUT "Enter word:"; A$
L = LEN(A$)
FOR I = 1 TO L - 1
  PRINT SPACE$(L - 1); MID$(A$, I, 1)
NEXT I
PRINT A$


'1.4
' This program prints a pattern of numbers in pyramid form.
'
INPUT "Enter N:"; N
FOR I = 1 TO N
  PRINT SPACE$(10 - I); : PRINT USING "#"; I;
  IF I > 1 THEN PRINT SPACE$(I * 2 - 3); : PRINT USING "#"; I;
  PRINT
NEXT I


'1.5
' This program corrects dates with A.D. or B.C.
'
INPUT "Enter date: "; D
INPUT "Enter A.D. or B.C.: "; A$
IF A$ = "B.C." AND D > 4 THEN PRINT D - 4; "B.C.": END
IF A$ = "B.C." THEN PRINT 5 - D; "A.D.": END
PRINT D + 4; "A.D"
```

```
'1.6
' This program will allow a user access with a password.
'
INPUT "ENTER PASSWORD:"; PSW$
I = 0
WHILE PSW$ <> "ITSME" AND I < 2
  PRINT "INVALID PASSWORD"
  INPUT "ENTER PASSWORD:"; PSW$
  I = I + 1
WEND
IF PSW$ = "ITSME" THEN
  PRINT "YOU HAVE ACCESS"
ELSE
  PRINT "YOU ARE TRESPASSING"
END IF


'1.7
' This program will display the best DBMS.
'
INPUT "Enter N: "; N:  MAX = 0
FOR I = 1 TO N
  INPUT "Enter DBMS name: "; D$
  INPUT "Enter convenience, efficiency:"; C, E
  IF C + E > MAX THEN MAX = C + E: NM$ = D$
NEXT I
PRINT NM$; " IS BEST"


'1.8
' This program displays the unique elements of a list.
'
INPUT "Enter #:"; N:  NUM = 0
WHILE N <> -999
  I = 1
  WHILE I <= NUM AND N <> A(I)
    I = I + 1
  WEND
  IF I > NUM THEN NUM = I: A(I) = N
  INPUT "Enter #:"; N
WEND
FOR I = 1 TO NUM:  PRINT LTRIM$(STR$(A(I))); " "; : NEXT I
PRINT


'1.9
' This program determines how many feet deep of dollar coins
' over Texas is equivalent to a given probability.
'
INPUT "Enter probability:"; PROB
DOLVOL = 1.5 * 1.5 * 3 / 32: TEXASAREA = 262134
TEXASVOL = TEXASAREA * 5280 * 12 * 5280 * 12
INCHDEEP = (PROB / (TEXASVOL / DOLVOL))
PRINT INT(INCHDEEP / 12 + .5); "FEET DEEP"
```

```
'1.10
' This program will map a logical address to the physical.
'
B(0) = 219:  L(0) = 600
B(1) = 2300: L(1) = 14
B(2) = 90:   L(2) = 100
B(3) = 1327: L(3) = 580
B(4) = 1952: L(4) = 96
INPUT "Enter Seg#, Address: "; S, A
WHILE S <= 4
  IF A > L(S) THEN
    PRINT "ADDRESSING ERROR"
  ELSE
    PRINT B(S) + A
  END IF
  INPUT "Enter Seg#, Address: "; S, A
WEND
```

```
'2.1
' This program prints F(x) for a recursive function given x.
'
INPUT "Enter x:"; X
F(1) = 1:  F(2) = 1:  F(3) = 1
I = 3
WHILE I < X
  F(I + 1) = (F(I) * F(I - 1) + 2) / F(I - 2)
  I = I + 1
WEND
PRINT "F("; : PRINT USING "#"; X; : PRINT ")="; F(X)


'2.2
' This program will print the prime factors of a number.
'
INPUT "Enter #:"; NUM
WHILE NUM > 1
  I = 2
  WHILE (NUM MOD I) > 0
    I = I + 1
  WEND
  PRINT I;
  NUM = INT(NUM / I)
  IF NUM > 1 THEN PRINT "X";
WEND


'2.3
' This program will display a word without its vowels.
'
INPUT "Enter word:"; WORD$
VOW$ = "AEIOU"
FOR I = 1 TO LEN(WORD$)
  CH$ = MID$(WORD$, I, 1)
  IF INSTR(VOW$, CH$) = 0 THEN PRINT CH$;
NEXT I
```

```
'2.4
' This program produces the shortest possible identifiers.
'
FOR I = 1 TO 6
  INPUT "Enter name: "; A$(I)
NEXT I
FOR I = 1 TO 6
  K = 1: S$ = LEFT$(A$(I), 1)
  FOR J = 1 TO 6
    WHILE (I <> J) AND S$ = MID$(A$(J), 1, K) AND (K < LEN(A$(I)))
      K = K + 1
      S$ = S$ + MID$(A$(I), K, 1)
    WEND
  NEXT J
  PRINT S$
NEXT I


'2.5
' This program prints the # of distinguishable permutations.
'
DIM LETTER(26)
INPUT "Enter word:"; WORD$:  L = LEN(WORD$)
' Calculate L factorial (assuming all different letters)
NUM = 1
FOR I = 1 TO L:  NUM = NUM * I:  NEXT I
' Divide out of Num the factorials of the same letters
FOR I = 1 TO L
  LETPOS = ASC(MID$(WORD$, I, 1)) - 64
  LETTER(LETPOS) = LETTER(LETPOS) + 1
  IF LETTER(LETPOS) > 1 THEN NUM = NUM / LETTER(LETPOS)
NEXT I
PRINT NUM


'2.6
' This program underlines parts of a sentence between 2 *'s.
'
INPUT "Enter sentence:"; SENT$
CLS : PRINT SENT$
UNDER = 0:  COL = 0
FOR I = 1 TO LEN(SENT$)
  CH$ = MID$(SENT$, I, 1)
  IF CH$ = "*" THEN
    UNDER = NOT UNDER
  ELSE
    COL = COL + 1
    LOCATE 3, COL: PRINT CH$
    IF UNDER THEN LOCATE 4, COL: PRINT "-"
  END IF
NEXT I
PRINT
```

```
'2.7
' This program will compute an expression containing + - * /.
'
INPUT "Enter expression:"; ST$:  NUMST$ = ""
' Parse first number in Num1 and second number in Num2
FOR I = 1 TO LEN(ST$)
  CH$ = MID$(ST$, I, 1)
  IF INSTR("+-*/", CH$) > 0 THEN
    SYMBOL$ = CH$:  NUM1 = VAL(NUMST$):  NUMST$ = ""
  ELSE
    NUMST$ = NUMST$ + CH$
  END IF
NEXT I
NUM2 = VAL(NUMST$)
IF SYMBOL$ = "+" THEN PRINT NUM1 + NUM2
IF SYMBOL$ = "-" THEN PRINT NUM1 - NUM2
IF SYMBOL$ = "*" THEN PRINT NUM1 * NUM2
IF SYMBOL$ = "/" THEN PRINT NUM1 / NUM2


'2.8
' This program will display the saddle point of a matrix.
'
DIM MAT(5, 5)
INPUT "Enter # Rows, # Cols:"; ROWS, COLS
FOR I = 1 TO ROWS
  FOR J = 1 TO COLS
    PRINT USING "Enter Row#"; I;
    PRINT USING " Col#"; J;
    INPUT MAT(I, J)
  NEXT J
NEXT I
' Find value smallest in row, largest in column
FOR I = 1 TO ROWS
  FOR J = 1 TO COLS
    SMALL = -1
    FOR K = 1 TO COLS
      IF (K <> J) AND (MAT(I, J) >= MAT(I, K)) THEN SMALL = 0
    NEXT K
    IF SMALL THEN
      LARGE = -1
      FOR K = 1 TO ROWS
        IF (K <> I) AND (MAT(I, J) <= MAT(K, J)) THEN LARGE = 0
      NEXT K
      IF LARGE THEN
        PRINT "SADDLE POINT ="; MAT(I, J); "AT ROW"; I;
        PRINT "COL"; J
      END IF
    END IF
  NEXT J
NEXT I
```

```
'2.9
' This program will sort a set of dates in increasing order.
'
DIM MO$(12)
DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY,AUGUST
DATA SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
FOR I = 1 TO 12:  READ MO$(I):  NEXT I
INPUT "Enter # of dates:"; N
FOR I = 1 TO N
  INPUT "Enter month:"; M$(I)
  INPUT "Enter day:  "; D(I)
  INPUT "Enter year: "; Y(I)
  PRINT
'  Combine year, month, day (in that order) for sorting
  J = 1
  WHILE (J < 13) AND (M$(I) <> MO$(J)):  J = J + 1:  WEND
  SORT(I) = ((Y(I) * 100) + J) * 100 + D(I)
  INDEX(I) = I
NEXT I
' Sort dates according to values in Sort() and swap index()
FOR I = 1 TO N - 1
  FOR J = I + 1 TO N
    IF SORT(INDEX(I)) > SORT(INDEX(J)) THEN
      SWAP INDEX(I), INDEX(J)
    END IF
  NEXT J
NEXT I
FOR I = 1 TO N
  PRINT M$(INDEX(I)); D(INDEX(I)); Y(INDEX(I))
NEXT I
```

```
'2.10
' This program displays class grades and the averages.
'
DIM QIZ(5, 4)
DATA "D. WOOLY","M. SMITH","C. BROWN","R. GREEN","T. STONE"
FOR I = 1 TO 5:  READ NAM$(I): NEXT I
DATA 100,92,90,90, 55,75,70,65, 94,70,62,70
DATA 90,74,80,85, 85,98,100,70
FOR I = 1 TO 5
  FOR J = 1 TO 4
    READ QIZ(I, J)
  NEXT J
NEXT I
FOR SCR = 1 TO 2
  CLS
  IF SCR = 2 THEN
    PRINT "           MS. HEINDEL'S MUSIC CLASS"
    PRINT "                  FINAL GRADES"
    PRINT "                  SPRING 1989"
    PRINT
  END IF
  PRINT "  NAME       Q1     Q2     Q3     Q4";
  IF SCR = 2 THEN PRINT "    AVERAGE" ELSE PRINT
  PRINT
'
  FOR I = 1 TO 5
    PRINT NAM$(I); : SUM = 0
    FOR J = 1 TO 4
      PRINT SPACE$(4); : PRINT USING "###"; QIZ(I, J);
      SUM = SUM + QIZ(I, J)
    NEXT J
    IF SCR = 2 THEN PRINT USING "   ###.##"; SUM / 4 ELSE PRINT
  NEXT I
  PRINT
  IF SCR = 1 THEN
    PRINT "Enter 5 grades for quiz 4:";
    INPUT QIZ(1, 4), QIZ(2, 4), QIZ(3, 4), QIZ(4, 4), QIZ(5, 4)
  END IF
NEXT SCR
' Display Column averages and class average
PRINT "AVERAGE:"; : TOTAL = 0
FOR I = 1 TO 4
  SUM = 0
  FOR J = 1 TO 5:  SUM = SUM + QIZ(J, I): NEXT J
  PRINT USING " ###.##"; SUM / 5;
  TOTAL = TOTAL + SUM
NEXT I
PRINT : PRINT
PRINT USING "CLASS AVERAGE:###.##"; TOTAL / 20
```

```
'3.1
' This program will determine if a word is correctly spelled.
'
INPUT "Enter word:"; ST$
L = LEN(ST$):  CORRECT = -1
'-- Check for E before suffixes ING, IBLE, ABLE
IF L >= 4 THEN
  PART$ = MID$(ST$, L - 2, 3)
  IF PART$ = "ING" AND MID$(ST$, L - 3, 1) = "E" THEN CORRECT = 0
END IF
IF L >= 5 THEN
  PART$ = MID$(ST$, L - 3, 4)
  IF PART$ = "IBLE" AND MID$(ST$, L - 4, 1) = "E" THEN CORRECT = 0
  IF PART$ = "ABLE" AND MID$(ST$, L - 4, 1) = "E" THEN CORRECT = 0
END IF
'-- Check if IE after C.
PART$ = ST$:  I = INSTR(PART$, "IE")
WHILE (I > 0) AND CORRECT
  I = I - 1
  IF I >= 1 THEN IF MID$(PART$, I, 1) = "C" THEN CORRECT = 0
  PART$ = MID$(PART$, I + 3, LEN(PART$) - (I + 2))
  I = INSTR(PART$, "IE")
WEND
'-- Check if EI not after C.
PART$ = ST$:  I = INSTR(PART$, "EI")
WHILE (I > 0) AND CORRECT
  CORRECT = 0
  IF I >= 2 THEN IF MID$(PART$, I - 1, 1) = "C" THEN CORRECT = -1
  PART$ = MID$(PART$, I + 3, LEN(PART$) - (I + 2))
  I = INSTR(PART$, "EI")
WEND
'-- Check for 3 consecutive same letters
I = 1
WHILE (I <= L - 2) AND CORRECT
  IF MID$(ST$, I, 1) = MID$(ST$, I + 1, 1) THEN
    IF MID$(ST$, I, 1) = MID$(ST$, I + 2, 1) THEN
      CORRECT = 0
    END IF
  END IF
  I = I + 1
WEND
IF CORRECT THEN PRINT "CORRECT" ELSE PRINT "MISSPELLED"
```

```
'3.2
' This program finds the positive root of V for an equation.
'
DEF FNC (V) = -23511.9 * V * V + 988686.1 * V - 400943!
DEF FNB (V) = P(I) * V * 9062.599
DEF FNA (V) = P(I) * V * V * V * 14.14 - FNB(V) + FNC(V)
DATA 0.05, 0.7, 10.0, 70.0
FOR I = 1 TO 4: READ P(I): NEXT I
FOR I = 1 TO 5
  IF I = 5 THEN PRINT : INPUT "Enter value for P:"; P(5)
  FOR J = 0 TO 2
    IF SGN(FNA(J)) <> SGN(FNA(J + 1)) AND FNA(J + 1) <> 0 THEN
      LOW = J: HIGH = J + 1
      IF FNA(LOW) > FNA(HIGH) THEN SWAP LOW, HIGH
      WHILE ABS(LOW - HIGH) > .00005
        MID = (LOW + HIGH) / 2
        IF FNA(MID) < 0 THEN LOW = MID ELSE HIGH = MID
      WEND
      MID = SGN(MID) * INT(ABS(MID) * 10000 + .5) / 10000
      PRINT USING "P = ##.##"; P(I);
      PRINT USING "  V = #.####"; MID
    END IF
  NEXT J
NEXT I
```

```
'3.3
' This program will magnify an input positive integer.
'
DATA 123567,36,13457,13467,2346,12467,124567,136,1234567,12346
FOR I = 0 TO 9:  READ NUM$(I):  NEXT I
INPUT "Enter number:"; N$
INPUT "Enter magnification:"; MAGN
CLS
FOR I = 1 TO LEN(N$)
  N = VAL(MID$(N$, I, 1))
  COL = (I - 1) * MAGN * 6 + 1
  FOR J = 1 TO LEN(NUM$(N))
     PART = VAL(MID$(NUM$(N), J, 1))
     GOSUB DisplayPart
  NEXT J
NEXT I
END
'
DisplayPart:
  SELECT CASE PART
    CASE 1
      LOCATE 1, COL
      FOR K = 1 TO MAGN: PRINT "****"; : NEXT K: PRINT
    CASE 2
      FOR K = 1 TO MAGN * 2 + 1: LOCATE K, COL: PRINT "*": NEXT K
    CASE 3
      FOR K = 1 TO MAGN * 2 + 1
        LOCATE K, COL + MAGN * 4 - 1: PRINT "*"
      NEXT K
    CASE 4
      LOCATE MAGN * 2 + 1, COL
      FOR K = 1 TO MAGN: PRINT "****"; : NEXT K: PRINT
    CASE 5
      FOR K = MAGN * 2 + 1 TO MAGN * 4 + 1
        LOCATE K, COL: PRINT "*"
      NEXT K
    CASE 6
      FOR K = MAGN * 2 + 1 TO MAGN * 4 + 1
        LOCATE K, COL + MAGN * 4 - 1: PRINT "*"
      NEXT K
    CASE 7
      LOCATE MAGN * 4 + 1, COL
      FOR K = 1 TO MAGN: PRINT "****"; : NEXT K: PRINT
  END SELECT
RETURN
```

```
'3.4
' This program produces a calendar for a given month/year.
' January 1, 1901 is a Tuesday.
'
DIM MO$(12), DAYSINMO(12)
DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY
DATA AUGUST,SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
FOR I = 1 TO 12: READ MO$(I): NEXT I
FOR I = 1 TO 12: READ DAYSINMO(I): NEXT I
INPUT "Enter month, year:"; MONTH, YEAR
MD = 2 + INT((26 - (LEN(MO$(MONTH)) + 5)) / 2)
CLS : PRINT SPACE$(MD); MO$(MONTH); YEAR
PRINT "   S   M   T   W   T   F   S"
PRINT "  -------------------------"
' Calculate # of days from 1/1/1901 to last day of prior month
DAYS = (YEAR - 1901) * 365 + INT((YEAR - 1901) / 4)
FOR I = 1 TO MONTH - 1
  DAYS = DAYS + DAYSINMO(I)
NEXT I
IF (MONTH > 2) AND (YEAR MOD 4 = 0) THEN DAYS = DAYS + 1
' Determine first day of month
DAY = (DAYS + 1) MOD 7  'Day =0 (Mon), =1 (Tue) ... =6 (Sun)
COL = (DAY + 1) MOD 7   ' Day = 0,1,2,3,4,5,6 Sun,Mon...Sat
IF (MONTH = 2) AND (YEAR MOD 4 = 0) THEN LEAP = 1 ELSE LEAP = 0
' Display month calendar
IF COL > 0 THEN PRINT SPACE$(COL * 4);
FOR I = 1 TO DAYSINMO(MONTH) + LEAP
  PRINT USING "####"; I;
  COL = (COL + 1) MOD 7
  IF COL = 0 THEN PRINT
NEXT I
```

```
'3.5
' This program positions 5 queens on the board so none attack.
'
PRINT "ROWS = 1 2 3 4 5"
PRINT "----------------"
PRINT "COLUMNS"
COL = 1:  ROW = 1: DIMEN = 5
WHILE (COL > 1) OR (ROW < DIMEN + 1)
  WHILE (ROW <= DIMEN) AND (COL <= DIMEN)
    GOSUB IsQueenSafe
    IF SAFETY THEN
      CONFIG(COL) = ROW:  COL = COL + 1:  ROW = 1
    ELSE
      ROW = ROW + 1
    END IF
  WEND
  IF (ROW = DIMEN + 1) THEN COL = COL - 1: ROW = CONFIG(COL) + 1
  IF (COL = DIMEN + 1) THEN
'   Display solution and retreat column
    PRINT SPACE$(6);
    FOR I = 1 TO DIMEN: PRINT USING "##"; CONFIG(I); : NEXT I
    PRINT
    COL = COL - 1: ROW = CONFIG(COL) + 1
  END IF
WEND
END
' -------- Function Safety returns True if no queen can attack
IsQueenSafe:
  SAFETY = -1
  FOR I = 1 TO COL - 1
    IF (CONFIG(I) + I) = (ROW + COL) THEN SAFETY = 0
    IF (CONFIG(I) - I) = (ROW - COL) THEN SAFETY = 0
    IF (CONFIG(I) = ROW) THEN SAFETY = 0
  NEXT I
  RETURN
```

```
'3.6
' This program prints the product of 2 large integers in Base.
'
DEFINT A-Z
DIM A(31), B(31), PROD(61)
INPUT "Enter base:"; BAS
INPUT "Enter first integer: "; ASTR$
INPUT "Enter second integer:"; BSTR$
' -- Determine if signs are positive or negative
SIGN = 1
IF MID$(ASTR$, 1, 1) = "-" THEN
  ASTR$ = MID$(ASTR$, 2, LEN(ASTR$) - 1): SIGN = -1
END IF
IF MID$(BSTR$, 1, 1) = "-" THEN
  BSTR$ = MID$(BSTR$, 2, LEN(BSTR$) - 1): SIGN = SIGN * -1
END IF
IF SIGN < 0 THEN PRINT "-";
' -- Store sgring digits into numerical arrays
LENA = LEN(ASTR$):  LENB = LEN(BSTR$)
FOR I = LENA TO 1 STEP -1
  A(LENA - I + 1) = VAL(MID$(ASTR$, I, 1))
NEXT I
FOR I = LENB TO 1 STEP -1
  B(LENB - I + 1) = VAL(MID$(BSTR$, I, 1))
NEXT I
' -- Multiply 2 numbers as a person would, with carries
FOR I = 1 TO LENB
  CARRY = 0
  FOR J = 1 TO LENA
    S = I + J - 1
    PROD(S) = PROD(S) + B(I) * A(J) + CARRY
    CARRY = INT(PROD(S) / BAS)
        PROD(S) = PROD(S) - CARRY * BAS
  NEXT J
  IF CARRY > 0 THEN PROD(S + 1) = CARRY
NEXT I
' -- Display product
IF CARRY > 0 THEN PRINT USING "#"; PROD(S + 1);
FOR I = S TO 1 STEP -1:  PRINT USING "#"; PROD(I); : NEXT I



'3.7
' This program computes most efficient change without a coin.
'
INPUT "Enter cost, amount:"; COST, AMOUNT
INPUT "Enter missing coin:"; COIN$
CHANGE = INT((AMOUNT - COST) * 100 + .1)
C$(1) = "QUARTER": C$(2) = "DIME": C$(3) = "NICKEL": C$(4) =
"PENNY"
A(1) = 25: A(2) = 10: A(3) = 5: A(4) = 1
X = CHANGE
ST = 1: EN = 4: GOSUB MakeChange   'Calculate denominations
C = 1
```

```
WHILE (C < 4) AND COIN$ <> C$(C): C = C + 1: WEND
SELECT CASE C
  CASE 1
'     ***  NO quarters  ***
'     Determine most efficient way without quarters (C=1)
    X = CHANGE
    ST = 2: EN = 4: GOSUB MakeChange   'Calculate denominations
  CASE 2
'     ***  NO dimes     ***
'     Add 2 nickels for every dime
    B(3) = B(3) + B(2) * 2
  CASE 3
'   ***  NO nickels   ***
'   IF a nickel then IF at least 1 quarter then
'                       Make 3 dimes and 1 less quarter
'                    Else make 5 more pennies with the 1 nickel
    IF B(3) = 1 THEN
      IF B(1) > 0 THEN
        B(2) = B(2) + 3: B(1) = B(1) - 1
      ELSE
        B(4) = B(4) + 5
      END IF
    END IF
END SELECT
'
' Display results
'
FOR I = 4 TO 1 STEP -1
   IF I <> C THEN
     PRINT USING "# "; B(I);
     IF I = 4 AND B(I) <> 1 THEN
       PRINT "PENNIES"
     ELSE
       PRINT C$(I); : IF B(I) <> 1 THEN PRINT "S" ELSE PRINT
     END IF
   END IF
NEXT I
PRINT "TOTAL CHANGE RETURNED ="; CHANGE; "CENT";
IF CHANGE <> 1 THEN PRINT "S" ELSE PRINT
END
'
' Determine most efficient change given coins
'
MakeChange:
  FOR I = ST TO EN
    B(I) = INT(X / A(I))
    X = X - B(I) * A(I)
  NEXT I
  RETURN
```

```
'3.8
' This program displays the coordinates of binary rectangles.
'
DEFINT A-Z
DIM A(6, 7)
' Convert 6 numbers to binary representation
FOR I = 1 TO 6
  INPUT "Enter number:"; NUM
  DEN = 128
  FOR J = 6 TO 0 STEP -1
    DEN = DEN / 2
    A(I, 7 - J) = INT(NUM / DEN)
    NUM = NUM - A(I, 7 - J) * DEN
  NEXT J
NEXT I
PRINT
' Display the 6 row X 7 col grid of 0s and 1s
FOR I = 1 TO 6
  FOR J = 1 TO 7
    PRINT USING "#"; A(I, J);
  NEXT J: PRINT
NEXT I
PRINT
' Find largest solid rectangles of 1s
FOR ROWLEN = 6 TO 2 STEP -1
  FOR COLLEN = 7 TO 2 STEP -1
    FOR ROWST = 1 TO 7 - ROWLEN
      FOR COLST = 1 TO 8 - COLLEN
        RECT = -1
        FOR I = ROWST TO ROWST + ROWLEN - 1
          J = COLST
          WHILE (J <= COLST + COLLEN - 1) AND RECT
            IF A(I, J) = 0 THEN RECT = 0
            J = J + 1
          WEND
        NEXT I
        IF RECT THEN
          PRINT USING "(#"; ROWST; : PRINT ",";
          PRINT USING "#"; COLST; :  PRINT ")";
          PRINT USING "(#"; ROWST + ROWLEN - 1; : PRINT ",";
          PRINT USING "#"; COLST + COLLEN - 1; :  PRINT ")"
          FOR I = ROWST TO ROWST + ROWLEN - 1
            FOR J = COLST TO COLST + COLLEN - 1
              A(I, J) = 0
            NEXT J
          NEXT I
        END IF
      NEXT COLST
    NEXT ROWST
  NEXT COLLEN
NEXT ROWLEN
```

```
'3.9
' This program determines the 5 word combination for BINGO.
'
DIM LETVAL(26)
DATA 9, 14, 1, 16, 20, 5, 10, 2, 21, 17, 6, 25
DATA 12, 3, 22, 18, 24, 7, 13, 26, 15, 11, 19, 4, 23, 8
FOR I = 1 TO 26:  READ LETVAL(I):  NEXT I
DATA BIBLE,IDYLL,NOISE,GULLY,OBESE
DATA OBESE,TITHE,INLET,IGLOO,TOWER
FOR COL = 1 TO 2
  FOR ROW = 1 TO 5
    READ HIGHWORD$(ROW, COL): SUM = 0
    FOR I = 1 TO 5
      WORD$ = HIGHWORD$(ROW, COL)
      LETTER$ = MID$(WORD$, I, 1)
      SUM = SUM + LETVAL(ASC(LETTER$) - 64)
    NEXT I
    HIGHEST(ROW, COL) = SUM
  NEXT ROW
NEXT COL
'
WHILE WORD$ <> "QUIT"
  GOSUB DisplayValues   'DisplayValues
  INPUT "Enter word:"; WORD$
  WHILE LEN(WORD$) = 5
    SUM = 0
    FOR I = 1 TO 5
      LETTER$ = MID$(WORD$, I, 1)
      LETTERS$(I) = LETTER$
      SUM = SUM + LETVAL(ASC(LETTER$) - 64)
    NEXT I
    GOSUB UseWord
    INPUT "Enter word:"; WORD$
  WEND
WEND
END
'
'-- Procedure UseWord
UseWord:
  FOR COL = 1 TO 2
    FOR ROW = 1 TO 5
      IF LETTERS$(COL) = MID$("BINGO", ROW, 1) THEN
        IF SUM > HIGHEST(ROW, COL) THEN
          HIGHEST(ROW, COL) = SUM:  HIGHWORD$(ROW, COL) = WORD$
        END IF
      END IF
    NEXT ROW
  NEXT COL
  RETURN

'
'-- Procedure DisplayValues
DisplayValues:
  PRINT : MAX = 0
  FOR I = 1 TO 2:  MAXSUM(I) = 0:  NEXT I
```

```
  ST = 1:  EN = 2
  FOR ROW = 1 TO 5
    FOR COL = ST TO EN
      PRINT HIGHWORD$(ROW, COL);
      PRINT USING " ###"; HIGHEST(ROW, COL);
      PRINT SPACE$(3);
      MAXSUM(COL) = MAXSUM(COL) + HIGHEST(ROW, COL)
    NEXT COL
    PRINT
  NEXT ROW
' Determine maximum column and display ***
  FOR COL = ST TO EN
    PRINT SPACE$(3 + COL * 3); : PRINT USING "###"; MAXSUM(COL);
    IF MAXSUM(COL) > MAX THEN MAX = MAXSUM(COL): MAXCOL = COL
  NEXT COL
  PRINT
  IF MAXCOL = 1 THEN
    PRINT SPACE$(6); "***"
  ELSE
    PRINT SPACE$(18); "***"
  END IF
  PRINT
  RETURN



'3.10
' This program displays the number of distinguishable
' permutations for a cube w/sides input as color symbols.
'
DIM UNIQUE$(24, 6)
DATA TOP,FRONT,BOTTOM,BACK,RIGHT,LEFT
FOR I = 1 TO 6: READ SIDE$(I): NEXT I
' Assign colors to original 4 cubes
FOR I = 1 TO 6
  PRINT "Enter "; SIDE$(I); " side:"; : INPUT CUBE$(I)
NEXT I
NUM = 0
' Rotate cubes and check if it is unique
FOR ROT = 0 TO 23
  GOSUB Permute
  IF ROT = 0 THEN
    VALID = -1
  ELSE
    J = 1: VALID = -1
    WHILE (J <= NUM) AND VALID
      VALID = 0
      FOR K = 1 TO 6
        IF C$(K) <> UNIQUE$(J, K) THEN VALID = -1
      NEXT K
      J = J + 1
    WEND
  END IF
  IF VALID THEN
    NUM = NUM + 1
```

```
    FOR I = 1 TO 6: UNIQUE$(NUM, I) = C$(I): NEXT I
  END IF
NEXT ROT
PRINT "NUMBER OF DISTINGUISHABLE CUBES ="; NUM
END
'-- PROCEDURE THAT PERMUTES (SWAPS THE COLORS ON THE SQUARES)
Permute:
  IF ROT MOD 4 > 0 THEN
     TEMP$ = C$(2): C$(2) = C$(5): C$(5) = C$(4)
     C$(4) = C$(6): C$(6) = TEMP$
  ELSE
    SQUARE = INT(ROT / 4) + 1
    C$(1) = CUBE$(SQUARE)
    SELECT CASE SQUARE
      CASE 1
        FOR I = 2 TO 6: C$(I) = CUBE$(I): NEXT I
      CASE 2
        C$(2) = CUBE$(3): C$(3) = CUBE$(4)
        C$(4) = CUBE$(1): C$(5) = CUBE$(5): C$(6) = CUBE$(6)
      CASE 3
        C$(2) = CUBE$(4): C$(3) = CUBE$(1)
        C$(4) = CUBE$(2): C$(5) = CUBE$(5): C$(6) = CUBE$(6)
      CASE 4
        C$(2) = CUBE$(1): C$(3) = CUBE$(2)
        C$(4) = CUBE$(3): C$(5) = CUBE$(5): C$(6) = CUBE$(6)
      CASE 5
        C$(2) = CUBE$(2): C$(3) = CUBE$(6)
        C$(4) = CUBE$(4): C$(5) = CUBE$(3): C$(6) = CUBE$(1)
      CASE 6
        C$(2) = CUBE$(2): C$(3) = CUBE$(5)
        C$(4) = CUBE$(4): C$(5) = CUBE$(1): C$(6) = CUBE$(3)
    END SELECT
  END IF
  RETURN
```