```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '93 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T93;
{ -- This program displays six lines with "GTEDS". }
{ -- The solution could also be done with 6 Writeln statements. }
  var
    I, J: Byte;

begin
  for I := 1 to 6 do begin
    for J := 1 to 7 - I do begin
      Write ('GTEDS', ' ':I);
    end;
    Writeln;
  end;
end.



{1.2}
program One2T93;
{ -- This program displays the number of programmers placed. }
  var
    N, M: Integer;

begin
  Write ('Enter N: ');  Readln (N);
  Write ('Enter M: ');  Readln (M);
  Writeln (N * 15 - M, ' PROGRAMMERS');
end.



{1.3}
program One3T93;
{ -- This program will format the number N million with commas. }
  var
    N:   Real;
    NSt: String[12];

begin
  Write ('Enter N: ');  Readln (N);
  STR (N * 1E6 :9:0, NSt);
  Insert (',', NSt, 7);  Insert (',', NSt, 4);
  Writeln (NSt, ' ACCESS LINES');
end.
```

```
{1.4}
program One4T93;
{ -- This program will total # of students on 5 USF campuses. }
  const
    Campus: Array[1..5] of String[14] =  ('Tampa',
     'St. Petersburg', 'Fort Myers', 'Lakeland', 'Sarasota');
  var
    Num, Total: LongInt;
    I:          Byte;

begin
  Total := 0;
  for I := 1 to 5 do begin
    Write ('Enter # at ', Campus[I], ': ');  Readln (Num);
    Total := Total + Num;
  end;
  Write (Total, ' STUDENTS');
end.


{1.5}
program One5T93;
{ -- This program will determine if person qualifies for ISOP. }
  var
    Name:   String[12];
    Level:  Byte;
    Desire: String[3];

begin
  Write ('Enter Name: ');    Readln (Name);
  Write ('Enter level: ');   Readln (Level);
  Write ('Enter desire: ');  Readln (Desire);
  Write (Name, ' IS ');
  if (level < 5) or (Desire = 'NO') then
    Write ('NOT ');
  Writeln ('A POSSIBLE CANDIDATE FOR ISOP');
end.
```

```
{1.6}
program One6T93;
{ -- This program will display preferred skills for curriculum. }
  var
    Curr: String[15];

begin
  Write ('Enter curriculum: ');  Readln (Curr);
  if Curr = 'MVS/COBOL' then
    begin
      Writeln ('COBOL');
      Writeln ('JCL');
      Writeln ('MVS/ESA');
      Writeln ('TSO/ISPF');
      Writeln ('VSAM');
      Writeln ('ANSI SQL');
      Writeln ('DB2');
      Writeln ('IMS');
    end
  else     { -- Curr = 'C/UNIX' }
    begin
      Writeln ('C');
      Writeln ('UNIX');
      Writeln ('ANSI SQL');
      Writeln ('OSF/MOTIF');
      Writeln ('SHELL PROGRAMMING');
    end;
end.


{1.7}
program One7T93;
{ -- This program will print the first N letters of alphabet. }
  var
    I, N: Byte;

begin
  Write ('Enter N: ');  Readln(N);
  for I := 1 to N do
    Write (Chr(64 + I));
end.
```

```
{1.8}
program One8T93;
{ -- This program will calculate the increase in salary. }
  var
    Salary, Increase: Real;
    Rating:           String[13];

begin
  Write ('Enter salary: ');  Readln (Salary);
  Write ('Enter rating: ');  Readln (Rating);
  if Rating = 'EXCELLENT' then
    Increase := Salary * 0.10
  else if Rating = 'ABOVE AVERAGE' then
    Increase := Salary * 0.07
  else if Rating = 'GOOD' then
    Increase := Salary * 0.05
  else
    Increase := 0.0;
  Writeln ('NEW SALARY = $', Salary + Increase: 7:2);
end.



{1.9}
program One9T93;
{ -- This program will display a Service Order. }
  var
    SO: String[7];
    Ch: Char;

begin
  Write ('Enter order: ');  Readln (SO);
  Ch := SO[1];
  if Length(SO) > 1 then
    Writeln (Ch)
  else
    Case Ch of
      'I':  Writeln ('INSTALL');
      'C':  Writeln ('CHANGE');
      'R':  Writeln ('RECORDS');
      'O':  Writeln ('OUT');
      'F':  Writeln ('FROM');
      'T':  Writeln ('TO');
    end;
end.
```

```
{1.10}
program One10T93;
{ -- This program will compute a GPA for 5 classes. }
  var
    G:           Char;
    I, Num, Sum: Byte;

begin
  Sum := 0;  Num := 5;
  for I := 1 to 5 do begin
    Write ('Enter grade: ');  Readln (G);
    case G of
      'A': Sum := Sum + 4;
      'B': Sum := Sum + 3;
      'C': Sum := Sum + 2;
      'D': Sum := Sum + 1;
    end;
    if G = 'W' then Num := Num - 1;
  end;
  Writeln ('GPA = ', Sum / Num : 4:3);
end.
```

```
{2.1}
program Two1T93;
{ -- This program will randomly generate #s between X and Y. }
  var
    I, N, X, Y, Min, Max: ShortInt;

begin
  Randomize;
  Write ('Enter N: ');  Readln (N);
  Write ('Enter X, Y: ');  Readln (X, Y);
  if X < Y then begin
    Min := X;  Max := Y;  end
  else begin
    Min := Y;  Max := X;
  end;

  for I := 1 to N do
    Write (Random(Max - Min + 1) + Min, ' ');

end.
```

```
{2.2}
program Two2T93;
{ -- This program will sort names according to their title. }
  const
    Titles: Array[1..7] of String[4] =
      ('P', 'PA', 'SA', 'SE', 'SSE', 'ASE', 'SASE');
  var
    Name:       Array [1..10] of String[20];
    Level:      Array [1..10] of Integer;
    Title:      String[4];
    TempN:      String[12];
    I, J, N, T: Byte;

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter name: ');  Readln (Name[I]);
    Write ('Enter title: '); Readln (Title);
    Name[I] := Name[I] + ' - ' + Title;
    J := 1;
    while Titles[J] <> Title do J := J + 1;
    Level[I] := J;
  end;

  for I := 1 to N - 1 do
    for J := I + 1 to N do
      if (Level[I] < Level[J])
      or ((Level[I] = Level[J]) and (Name[I] > Name[J])) then
       begin
        TempN := Name[I]; Name[I] := Name[J];   Name[J] := TempN;
        T := Level[I];   Level[I] := Level[J]; Level[J] := T;
       end;

  for I := 1 to N do
    Writeln (Name[I]);
end.
```

```
{2.3}
program Two3T93;
{ -- This program will format a COBOL declaration. }
  var
    Field:            Array[1..15] of String[30];
    Level, PrevLevel: String[2];
    I, J, Inc:        Integer;

begin
  I := 0;
  repeat
    I := I + 1;
    Write ('Enter field: ');  Readln (Field[I]);
  until Field[I] = '';

  for J := 1 to I - 1 do begin
    Level := Copy(Field[J], 1, 2);
    if Level = '01' then
      Inc := 0
    else if Level > PrevLevel then
      Inc := Inc + 4
    else if Level < PrevLevel then
      Inc := Inc - 4;

    if Inc > 0 then
      Write (' ': Inc);
    Writeln (Field[J]);

    PrevLevel := Level;
  end;  { -- for J }
end.
```

```
{2.4}
program Two4T93;
{ -- This program will translate a word and calculate blocks. }
  var
    Word, Number:          String[30];
    I, Num, Blocks, Code: Integer;
    Digit, LastDigit:     Byte;
    NumSt:                 String[2];

begin
  Write ('Enter word: ');  Readln (Word);

  Number := '';
  for I := 1 to Length(Word) do begin
    Num := Ord(Word[I]) - Ord('A') + 1;
    Str(Num, NumSt);
    Number := Number + NumSt;
  end;
  Writeln ('NUMBER = ', Number);

  Blocks := 1;
  Val (Copy(Number,1,1), LastDigit, Code);
  for I := 2 to Length(Number) do begin
    Val (Copy(Number,I,1), Digit, Code);
    if Digit mod 2 <> LastDigit mod 2 then
      Blocks := Blocks + 1;
    LastDigit := Digit;
  end;

  Writeln ('BLOCKS = ', Blocks);
end.
```

```
{2.5}
program Two5T93;
{ -- This program will display N formatted telephone #s. }
  var
    Num:          Array[1..15] of String[10];
    Line:         String[4];
    I, N, Total: Byte;
    NPA, NXX, NextNPA, NextNXX: String[3];

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter #: ');  Readln (Num[I]);
  end;
  Total := 1;  Num[I+1] := '     ';
  for I := 1 to N do begin
    NPA := Copy(Num[I], 1, 3);
    NXX := Copy(Num[I], 4, 3);
    Line:= Copy(Num[I], 7, 4);
    Write (NPA, '-', NXX, '-', Line);
    NextNPA := Copy(Num[I+1], 1, 3);
    NextNXX := Copy(Num[I+1], 4, 3);
    if (NPA <> NextNPA) then begin
      Writeln ('  TOTAL FOR NPA OF ', NPA, ' = ', Total);
      Writeln;
      Total := 1;
      end
    else begin
      Inc(Total);
      if NXX <> NextNXX then
        Writeln;
    end;
    Writeln;
  end;  { -- for I }
end.
```

```
{2.6}
program Two6T93;
{ -- This program will calculate product bought minus coupons. }
  var
    Prod, Coup:      Array[1..10] of String[1];
    Pric, Disc:      Array[1..10] of Real;
    Total, MaxDisc:  Real;
    I, J, NumProd, NumCoup, Ind: Byte;

begin
  I := 0;
  repeat
    Inc(I);
    Write ('Enter product: '); Readln (Prod[I]);
    if Prod[I] <> '9' then begin
      Write ('Enter price: ');  Readln (Pric[I]);
    end;
  until Prod[I] = '9';
  NumProd := I - 1;
  Writeln;

  J := 0;
  repeat
    Inc(J);
    Write ('Enter coupon: '); Readln (Coup[J]);
    if Coup[J] <> '9' then begin
      Write ('Enter discount: ');  Readln (Disc[J]);
    end;
  until Coup[J] = '9';
  NumCoup := J - 1;

  Total := 0;
  for I := 1 to NumProd do begin
    MaxDisc := 0;
    for J := 1 to NumCoup do
      if Prod[I] = Coup[J] then
        if Disc[J] > MaxDisc then begin
          MaxDisc := Disc[J];  Ind := J;
        end;
    Total := Total + Pric[I] - MaxDisc;
    Coup[Ind] := '*';
  end;

  Writeln;
  Writeln ('TOTAL = $', Total: 4:2);
end.
```

```
{2.7}
program Two7T93;
{ -- This program will display dates in other formats. }
  var
    Format: String[8];
    Date:   String[10];
    YYYY:   String[4];
    DD, MM: String[2];

begin
  Write ('Enter format: ');  Readln (Format);
  Write ('Enter date: ');    Readln (Date);
  if Format = 'ISO' then begin
    YYYY := Copy (Date, 1, 4);
    MM   := Copy (Date, 6, 2);
    DD   := Copy (Date, 9, 2);
    end
  else if Format = 'AMERICAN' then begin
    MM   := Copy (Date, 1, 2);
    DD   := Copy (Date, 4, 2);
    YYYY := Copy (Date, 7, 4);
    end
  else begin  { -- Format = 'EUROPEAN' }
    DD   := Copy (Date, 1, 2);
    MM   := Copy (Date, 4, 2);
    YYYY := Copy (Date, 7, 4);
  end;

  if Format <> 'ISO' then
    Writeln ('ISO = ', YYYY, '-', MM, '-', DD);
  if Format <> 'AMERICAN' then
    Writeln ('AMERICAN = ', MM, '-', DD, '-', YYYY);
  if Format <> 'EUROPEAN' then
    Writeln ('EUROPEAN = ', DD, '-', MM, '-', YYYY);
end.
```

```
{2.8}
program Two8T93;
{ -- This program will reverse the words in 1 or 2 sentences. }
  var
    Sent:      String;
    Word:      Array [1..10] of String[15];
    I, J, Num: Integer;
    Ch:        Char;

begin
  Write ('Enter sentence: ');  Readln (Sent);
  Num := 1;  Word[Num] := '';  I := 1;
  repeat
    Ch := Sent[I];
    if Ch = '.' then
      begin
        for J := Num downto 1 do
          if J = Num then
            Write (Word[J])
          else
            Write (' ', Word[J]);
        Write ('.  ');
        Num := 0;  Inc(I);
      end
    else
      if Ch <> ' ' then  { -- Add letter to word. }
        Word[Num] := Word[Num] + Ch
      else  { -- Word completed by a space. }
        begin
          Inc(Num);
          Word[Num] := '';
        end;
    Inc(I);
  until (I > Length(Sent));
end.
```

```
{2.9}
program Two9T93;
{ -- This program will print 4 smallest #s in a 4 x 4 matrix. }
  var
    I, J, K, X, Num: Byte;
    A: Array [1..4, 1..4] of ShortInt;
    B: Array [0..16] of ShortInt;
    OneDisplayed: Boolean;

begin
  for I := 1 to 4 do begin
    Write ('Enter row ', I, ': ');
    Readln (A[I,1], A[I,2], A[I,3], A[I,4]);
  end;

  for I := 1 to 4 do
    for J := 1 to 4 do
      B[(I - 1) * 4 + J] := A[I,J];

  for I := 1 to 15 do
    for J := I + 1 to 16 do
      if B[I] > B[J] then begin
        X := B[I];  B[I] := B[J];  B[J] := X;
      end;

  K := 1;  Num := 0;  B[0] := -99;
  repeat
    OneDisplayed := False;
    if B[K] <> B[K-1] then begin
      Writeln;
      Inc(Num);
      Write (Num, '. SMALLEST = ', B[K], ' OCCURS AT ');
      for I := 1 to 4 do
        for J := 1 to 4 do
          if B[K] = A[I,J] then begin
            if OneDisplayed then
              Write (', ')
            else
              OneDisplayed := True;
            Write ('(', I, ',', J, ')');
          end;
    end;  { -- if B[K] }
    Inc(K);
  until (Num = 4) and (B[K] <> B[K-1]);
end.
```

```
{2.10}
program Two10T93;
{ -- This program will print # of days between two dates. }
  const
    Month: Array [1..12] of Byte =
      (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
  var
    M, D, Y, I, Days, Days2: Integer;

begin
  Write ('Enter month: ');  Readln (M);
  Write ('Enter day: ');    Readln (D);
  Write ('Enter year: ');   Readln (Y);
  Days := 0;  Days2 := 0;
  { -- October 25, 1967 }
  for I := 1 to 9 do
    Days2 := Days2 + Month[I];
  Days2 := Days2 + 25;

  for I := 1967 to Y - 1 do begin
    Days := Days + 365;
    if I mod 4 = 0 then Days := Days + 1;
  end;
  if (Y mod 4 = 0) and (M > 2) then Days := Days + 1;
  for I := 1 to M - 1 do
    Days := Days + Month[I];
  Days := Days + D;

  Writeln (Days - Days2, ' DAYS');
end.
```

```
{3.1}
program Thr1T93;
{ -- This program displays GTEDS squares relative to cursor. }
{ -- Cursor can be moved up, left, down, right: I, J, K, M. }
uses Crt;
  var
    R, C, X, A, B: Integer;
    K: Char;

begin
  ClrScr;  R := 5;  C := 5;  K := ' ';
  while not (K in ['1' .. '4']) do begin
    GotoXY (C, R);  Write ('#');  K := ' ';
    K := ReadKey;
    if K in ['I', 'J', 'K', 'M'] then begin
      GotoXY (C, R);  Write (' ');
      if K = 'I' then Dec(R);
      if K = 'M' then Inc(R);
      if K = 'J' then Dec(C);
      if K = 'K' then Inc(C);
    end;
  end;
  X := Ord(K) - Ord('0');
  if X = 1 then begin  A := 1;  B :=  0;  end;
  if X = 2 then begin  A := 1;  B := -1;  end;
  if X = 3 then begin  A := -1; B := -1;  end;
  if X = 4 then begin  A := -1; B := 0;   end;
  if (R + 5*A > 24)     or (R + 5*A < 1) or
     (C + 9*B + 9 > 80) or (C + 9*B < 1) then
    begin
       ClrScr;  Writeln ('OFF THE SCREEN');
    end
  else
    begin
      GotoXY (C + 8*B, R + 1*A);  Writeln ('G T E D S');
      GotoXY (C + 8*B, R + 2*A);  Writeln ('T       D');
      GotoXY (C + 8*B, R + 3*A);  Writeln ('E   ', X, '   E');
      GotoXY (C + 8*B, R + 4*A);  Writeln ('D       T');
      GotoXY (C + 8*B, R + 5*A);  Writeln ('S D E T G');
    end;
end.
```

```
{3.2}
program Thr2T93;
{ -- This program will solve an equation with +,-,*, or /. }
  var
    V1, V2, V3, S1, S2, X:  String[3];
    N1, N2, N3, I, J, Code: Integer;

begin
  Write ('Enter value: ');  Readln (V1);
  Write ('Enter symbol: '); Readln (S1);
  Write ('Enter value: ');  Readln (V2);
  Write ('Enter symbol: '); Readln (S2);
  Write ('Enter value: ');  Readln (V3);
  if S1 = '=' then begin
    S1 := S2;  S2 := '=';
    X := V1;   V1 := V2;  V2 := V3;  V3 := X;
  end;
  { -- Equation is now of the form: V1 [op] V2 = V3 }
  Val(V1, N1, Code);
  Val(V2, N2, Code);
  Val(V3, N3, Code);
  Write ('X = ');
  if S1 = '+' then
    if V1 = 'X' then
      Writeln (N3 - N2)
    else if V2 = 'X' then
      Writeln (N3 - N1)
    else
      Writeln (N1 + N2);

  if S1 = '-' then
    if V1 = 'X' then
      Writeln (N3 + N2)
    else if V2 = 'X' then
      Writeln (N1 - N3)
    else
      Writeln (N1 - N2);

  if S1 = '*' then
    if V1 = 'X' then
      Writeln (N3 div N2)
    else if V2 = 'X' then
      Writeln (N3 div N1)
    else
      Writeln (N1 * N2);

  if S1 = '/' then
    if V1 = 'X' then
      Writeln (N3 * N2)
    else if V2 = 'X' then
      Writeln (N1 div N3)
    else
      Writeln (N1 div N2);
end.
```

```
{3.3}
program Thr3T93;
{ -- This program prints combinations of digits summing to #. }
  var
    Digits:     String[7];
    Digit, A:   Array[1..7] of Byte;
    OneWritten: Boolean;
    I, J, Sum, NewSum, Code, Last, Total, Power: Integer;

begin
  Write ('Enter digits: ');  Readln (Digits);
  Write ('Enter sum: ');  Readln (Sum);
  NewSum := (Sum div 10) * 8 + (Sum mod 10);
  Last := Length(Digits);
  for I := 1 to Last do
    Val(Copy(Digits, I, 1), Digit[I], Code);
  for I := 1 to Last do
    A[I] := 0;

  Power := 1;
  for I := 1 to Last do Power := Power * 2;
  Power := Power - 1;

  for I := 1 to Power do begin
    J := 1;
    while (A[J] = 1) do begin
      A[J] := 0;
      Inc(J);
    end;
    A[J] := 1;
    Total := 0;
    for J := 1 to Last do
      if A[J] = 1 then
        Total := Total + Digit[J];
    if Total = NewSum then begin
      OneWritten := False;
      for J := 1 to Last do
        if A[J] = 1 then
          if OneWritten then
            Write ('+', Digit[J])
          else begin
            Write (Digit[J]);
            OneWritten := True;
          end;
      Writeln (' = ', Sum);
    end;  { -- if }
  end;  { -- for I }
end.
```

```
{3.4}
program Thr4T93;
{ -- This program will decompose a large integer into primes. }
  var
    A, Q:    Array[1..80] of Integer;
    LongNum: String[80];
    Prime, I, J, L, Num, Power, Code:  Integer;
    IsPrime, FirstFactor, QuotientIs0: Boolean;

procedure DisplayFactor;
{ -- This procedure will display a Factor raised to a power. }
  begin
    if FirstFactor then
      FirstFactor := False
    else
      Write(' * ');
    Write (Prime);
    if Power > 1 then
      Write ('^', Power);
    Power := 0;
  end;

procedure GetNextPrime;
{ -- This procedure will get the next prime to divide LongNum. }
  begin
    if Prime = 2 then
      Prime := 3
    else
      repeat
        Prime := Prime + 2;
        IsPrime := True;
        for J := 3 to Trunc(Sqrt(Prime)) do
          if Prime mod J = 0 then IsPrime := False;
      until IsPrime;
  end;

begin
  Write ('Enter number: ');  Readln (LongNum);
  L := Length(LongNum);
  for I := 1 to L do
    Val(Copy(LongNum, I, 1), A[I], Code);
  Prime := 2;  Power := 0;
  FirstFactor := True;  QuotientIs0 := False;
  repeat
    { -- Check if LongNum (Array A) is divisible by Prime. }
    Num := 0;
    for I := 1 to L do begin
      Num := Num * 10 + A[I];
      Q[I] := Num div Prime;
      Num := Num - Q[I] * Prime;
    end;
    if Num = 0 then  { -- Prime divided LongNum. }
      begin
        I := 1;
        while (Q[I] = 0) and (I <= L) do
```

```
          Inc(I);
        QuotientIs0 := (I = L) and (Q[L] = 1);
        L := L - I + 1;
        { -- Copy Quotient into array A to be divided again. }
        for J := 1 to L do
          A[J] := Q[J + I - 1];
        Inc(Power);
      end
    else    { -- Prime did not divide LongNum. }
      begin
        if Power >= 1 then
          DisplayFactor;
        GetNextPrime;
      end;  { -- else }
  until QuotientIs0;
  DisplayFactor;
end.



{3.5}
program Thr5T93;
{ -- This program will find words in 12 x 11 array of letters. }
  const
    LetRow: Array [1..12] of String[11] =
      ('DATAADFBAAM', 'JARBJCEDFOI', 'REAEEXEVDBC',
       'JESUSDEERNR', 'FABUUNMIEMO', 'LLMNSOIPTKC',
       'POQRSITRUOH', 'ABUVKWSXPPI', 'SOYZCPULMLP',
       'CCISABCDOAM', 'AEFGRHIJCRM', 'LKLETTEKSID');
   var
     I, J, L, Row, Col, FCol, LCol, FRow, LRow: Byte;
     LetCol: Array[1..11] of String[12];
     Word:   Array[1..2]  of String[12];

procedure DisplayCoordinates (FRow, FCol, LRow, LCol: Integer);
{ -- This procedure will display first and last letter coord. }
  begin
    Writeln ('FIRST LETTER: (', FRow: 2, ', ', FCol: 2, ')');
    Writeln ('LAST LETTER:  (', LRow: 2, ', ', LCol: 2, ')');
  end;

begin
  { -- String together the columns instead of Rows. }
  for I := 1 to 11 do begin
    LetCol[I] := '';
    for J := 1 to 12 do
      LetCol[I] := LetCol[I] + Copy(LetRow[J], I, 1);
  end;

  Write ('Enter word: ');  Readln (Word[1]);
  L := Length(Word[1]);

  { -- Reverse Word. }
  Word[2] := '';
  for I := 1 to L do
```

```
    Word[2] := Word[2] + Copy(Word[1], L - I + 1, 1);

  { -- Find words horizontally, (frontwards and backwards). }
  J := 0;
  repeat
    Inc(J);
    Row := 0;
    repeat
      Inc(Row);
      Col := Pos (Word[J], LetRow[Row]);
    until (Row = 12) or (Col > 0);
  until (Col > 0) or (J = 2);
  if Col > 0 then begin
    if J = 1 then begin
      FCol := 0;  LCol := L - 1;  end
    else begin
      FCol := L - 1;  LCol := 0;
    end;
    DisplayCoordinates (Row, Col + FCol, Row, Col + LCol);
    Exit;
  end;

  { -- Find words vertically, (frontwards and backwards). }
  J := 0;
  repeat
    Inc(J);
    Col := 0;
    repeat
      Inc(Col);
      Row := Pos (Word[J], LetCol[Col]);
    until (Col = 11) or (Row > 0);
  until (Row > 0) or (J = 2);
  if Row > 0 then begin
    if J = 1 then begin
      FRow := 0;  LRow := L - 1;  end
    else begin
      FRow := L - 1;  LRow := 0;
    end;
    DisplayCoordinates (Row + FRow, Col, Row + LRow, Col);
    Exit;
  end;
end.
```

```
{3.6}
program Thr6T93;
{ -- This program will solve two inequality equations. }
  var
    Eq1, Eq2, Op: String[3];
    S1, S2:       String[1];
    N1, N2, Code, Min, Max: Integer;

procedure Display (X, Y: Integer);
{ -- This procedure will display all integers between X and Y. }
  var
    I: Integer;

  begin
    Write (X);
    for I := X + 1 to Y do Write (',', I);
  end;

begin
  Write ('Enter equation 1: ');  Readln (Eq1);
  Write ('Enter logical op: ');  Readln (Op);
  Write ('Enter equation 2: ');  Readln (Eq2);
  S1 := Copy(Eq1, 2, 1);
  S2 := Copy(Eq2, 2, 1);
  Val (Copy(Eq1, 3, 1), N1, Code);
  Val (Copy(Eq2, 3, 1), N2, Code);

  if (S1 = '<') and (S2 = '>') and (Op = 'AND') and (N1 <= N2)
  or (S1 = '>') and (S2 = '<') and (Op = 'AND') and (N1 >= N2)
  then
    Writeln ('NO SOLUTION');

  if (S1 = '<') and (S2 = '>') and (Op = 'OR') and (N1 > N2)
  or (S1 = '>') and (S2 = '<') and (Op = 'OR') and (N1 < N2)
  then
    Writeln ('ALL INTEGERS');

  if N1 < N2 then
    begin
      Min := N1;  Max := N2;
    end
  else
    begin
      Min := N2;  Max := N1;
    end;

  { -- Check for finite solution, and if less than 6 integers. }
  if (S1 = '<') and (S2 = '>') and (Op = 'AND') and (N1 > N2)
  or (S1 = '>') and (S2 = '<') and (Op = 'AND') and (N1 < N2)
  then
    if Max - Min <= 7 then
      Display (Min + 1, Max - 1)
    else begin
      Display (Min + 1, Min + 3);
      Write ('...');
```

```
      Display (Max - 3, Max - 1);
    end;

  { -- Check for infinite # of negative solutions. }
  if (S1 = '<') and (S2 = '<') and (Op = 'AND') then begin
    Write ('...');
    Display (Min - 3, Min - 1);
  end;

  { -- Check for infinite # of positive solutions. }
  if (S1 = '>') and (S2 = '>') and (Op = 'AND') then begin
    Display (Max + 1, Max + 3);
    Write ('...');
  end;

  { -- Check for infinite # of positive and negitive solutions. }
  if (S1 = '>') and (S2 = '<') and (Op = 'OR') and (N1 > N2)
  or (S1 = '<') and (S2 = '>') and (Op = 'OR') and (N1 < N2) then
  begin
    Write ('...');
    Display (Min - 3, Min - 1);
    Write ('   ');
    Display (Max + 1, Max + 3);
    Write ('...');
  end;
end.


{3.7}
program Thr7T93;
{ -- This program will print the sum and product of 2 matrices. }
  const
    Base16: String[16] = '0123456789ABCDEF';
  var
    Mat:           Array[1..2, 1..3, 1..3] of LongInt;
    Sum, Prod:     LongInt;
    I, J, K, L, X: Integer;
    Num:           String[4];

procedure ConvertToBase16(N: LongInt);
{ -- This procedure will convert a Sum/Product element to B16. }
  var
    I, D:       Integer;
    Power:      LongInt;
    FirstDigit: Boolean;

begin
  Write (' ');
  FirstDigit := False;
  Power := 1;
  for I := 1 to 4 do Power := Power * 16;
  for I := 1 to 5 do begin
    D := Trunc(N / Power);
    if (D = 0) and not FirstDigit then
      Write(' ')
```

```
    else begin
      Write (Copy(Base16, D + 1, 1));
      FirstDigit := True;
    end;
    N := N - D * Power;
    Power := Power div 16;
  end;
end;


begin
  for I := 1 to 2 do begin
    for J := 1 to 3 do
      for K := 1 to 3 do begin
        Write ('Enter Mat', I, ' (', J, ',', K, '): ');
        Readln (Num);
        L := Length(Num);
        if L = 2 then
          Mat[I,J,K] := (Pos(Copy(Num,1,1), Base16) - 1) * 16
        else
          Mat[I,J,K] := 0;
        X := Pos(Copy(Num,L,1), Base16) - 1;
        Mat[I,J,K] := Mat[I,J,K] + X;
      end;
    Writeln;
  end;

  { -- Compute Sum }
  Write ('SUM =');
  for I := 1 to 3 do begin
    for J := 1 to 3 do begin
      Sum := Mat[1, I, J] + Mat[2, I, J];
      ConvertToBase16(Sum);
    end;
    Writeln;
    If I < 3 then Write (' ': 5);
  end;
  Writeln;

  { -- Compute Product }
  Write ('PRODUCT =');
  for I := 1 to 3 do begin
    for J := 1 to 3 do begin
      Prod := 0;
      for K := 1 to 3 do
        Prod := Prod + Mat[1, I, K] * Mat[2, K, J];
      ConvertToBase16(Prod);
    end;
    Writeln;
    if I < 3 then Write (' ': 9);
  end;
end.
```

```
{3.8}
program Thr8T93;
{ -- This program will find three 3-digit primes. }
  var
    P:          Array [1..200] of Integer;
    A:          Array [1..9]   of Integer;
    P1, P2, P3: String[3];
    PCat:       String[9];
    I, J, K, L, Num, Pnum, Sq, Sum,
    X, Tot, D1, D2, D3, D4, N2, Code: Integer;

begin
  Num := 101;  Pnum := 0;
  repeat
    Sq := Trunc(Sqrt(Num));
    I := 1;
    repeat
      I := I + 2;
    until (I > Sq) or (Num mod I = 0);
    if (I > Sq) then begin
      N2 := Num;
      D1 := N2 div 100;
      N2 := N2 - D1 * 100;
      D2 := N2 div 10;
      D3 := N2 - D2 * 10;
      if not ((D1 = 0) or (D2 = 0) or (D3 = 0)
      or (D1 = D2) or (D2 = D3) or (D1 = D3)) then begin
        Pnum := Pnum + 1;
        P[Pnum] := Num;
      end;
    end;

    Num := Num + 2;
  until (Num > 999);

  for I := 1 to Pnum - 2 do
    for J := I + 1 to Pnum - 1 do
      for K := J + 1 to Pnum do begin
        Tot := P[I] + P[J] + P[K];
        if Tot > 1234 then begin
          Str (P[I], P1);
          Str (P[J], P2);
          Str (P[K], P3);
          PCat := P1 + P2 + P3;
          for L := 1 to 9 do A[L] := 0;
          L := 0;
          repeat
            L := L + 1;
            Val(Copy(PCat, L, 1), X, Code);
            A[X] := A[X] + 1;
          until (L = 9) or (A[X] = 2);
          if A[X] < 2 then begin
            Sum := Tot;
            D1 := Sum div 1000;
            Sum := Sum - D1 * 1000;
```

```
            D2 := Sum div 100;
            Sum := Sum - D2 * 100;
            D3 := Sum div 10;
            D4 := Sum - D3 * 10;
            if (D1 < D2) and (D2 < D3) and (D3 < D4) then begin
              Write (P[I], ' + ', P[J], ' + ', P[K], ' = ');
              Writeln (Tot);
            end;
          end;  { -- for K }
        end;  { -- for J }
      end;  { -- for I }

end.



{3.9}
program Thr9T93;
{ -- This program will produce a binary search tree. }
uses Crt;
  const
    ColInc: Array[0..8] of Byte =
     (0, 15, 7, 3, 1, 0, 0, 0, 0);
  var
    Words: String[50];
    A:     Array[0..8, 1..256] of String[1];
    Ch:    String[1];
    I, J, R, C, Col, PrevCol: Integer;


begin
  Write ('Enter word(s): ');  Readln (Words);
  { -- Initialize tree to nulls. }
  for I := 0 to 8 do
    for J := 1 to 256 do
      A[I, J] := '';
  ClrScr;

  for I := 1 to Length(Words) do begin
    Ch := Copy (Words, I, 1);
    if Ch <> ' ' then begin
      R := 0;  C := 1;  Col := 40;
      { -- Traverse tree until an empty node exists. }
      while A[R, C] <> '' do begin
        if Ch <= A[R, C] then
          begin
            C := 2 * C - 1;
            Col := Col - ColInc[R + 1] - 1;
          end
        else
          begin
            C := 2 * C;
            PrevCol := Col;
            Col := Col + ColInc[R + 1] + 1;
```

```
          end;
        R := R + 1;
      end;  { -- While }
      A[R, C] := Ch;

      GotoXY(Col, R + 1);
      if R = 0 then  { -- Place first letter in center. }
        Write (Ch)
      else
        if C mod 2 = 1 then  { -- Place letter right of parent. }
          begin
            Write (Ch);
            for J := 1 to ColInc[R] do Write ('-');
            Write ('+');
          end
        else
          begin  { -- Place letter left of parent. }
            GotoXY(PrevCol, R + 1);
            Write ('+');
            for J := 1 to ColInc[R] do Write ('-');
            Write (Ch);
          end;
    end;  { -- if Ch }
  end;  { -- for I }
end.



{3.10}
program Thr10T93;
{ -- This program will determine the values F(X) converges. }
  var
    K, Inc, Factor,
    FX, FX0, FX1, FX2: Real;
    F:                 Array[1..5000] of Real;
    I, X, Iter:        Integer;
    Diverge, Found:    Boolean;

begin
  K := 0;
  for I := 1 to 2 do begin
    if I = 1 then Inc := 0.01 else Inc := 0.1;
    Diverge := False;  Factor := 1;  Found := False;
    while (K < 10) and not Found do begin
      K := K + Inc / Factor;
      X := 1;  F[X] := K;
      if Factor < 20 then
        Iter := 250 * Trunc(Factor)
      else
        Iter := 5000;
      while (X < Iter) and not Diverge do begin
        X := X + 1;
        F[X] := Exp(Ln(K) * F[X - 1]);
        Diverge := (F[X] > 9.9);
      end;
```

```
      if I = 1 then
        begin
          FX2 := FX1;  FX1 := FX0;  FX0 := F[X];
          if (FX2 > FX1) and (FX1 < FX0) then begin
            K := K - 2 * Inc / Factor;
            if (FX2 - FX1) < 0.0005 then begin
              Found := True;  FX := FX1;
            end;
            FX0 := FX2;  FX1 := FX0;
            Factor := Factor * 2;
          end;
        end
      else  { -- I = 2 }
        if Diverge then
          begin
            Diverge := False;
            K := K - Inc / Factor;
            if Inc/ Factor < 0.000005 then Found := True;
            Factor := Factor * 2;
          end
        else
          FX := F[X];
    end;  { -- While }

    if I = 1 then Write ('MINIMUM') else Write ('MAXIMUM');
    Write (' VALUE: ');
    if I = 1 then
      begin
        Write   ('F(X) = ', FX : 4:3, ' OCCURS WHEN ');
        Writeln ('K = ', K + Inc / Factor :4:3);
      end
    else
      begin
        Write   ('F(X) = ', FX : 2:1, '   OCCURS WHEN ');
        Writeln ('K = ', K + Inc / Factor :6:5);
      end;
  end;  { -- for I }
end.
```