```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T95;
{ -- This program displays title of contest forward/backward. }
  const
    A: String[50] =
       'FLORIDA HIGH SCHOOLS COMPUTING COMPETITION ''95';
  var
    I, J: Integer;

begin
  for I := 1 to 4 do begin
    Writeln (A);
    for J := Length(A) downto 1 do
      Write(Copy(A, J, 1));
    Writeln;
  end;
end.


{1.2}
program One2T95;
{ -- This program generates comments in different languages. }
  var
    C: String[60];

begin
  Write ('Enter comment: ');  Readln (C);
  Writeln ('BASIC: '' ', C);
  Writeln ('PASCAL: { ', C, ' }');
  Writeln ('C: /* ', C, ' */');
  Writeln ('C++: // ', C);
end.


{1.3}
program One3T95;
{ -- This program either increments or decrements N by 1. }
  var
    N:  Integer;
    Op: String[2];

begin
  Write ('Enter N: ');  Readln (N);
  Write ('Enter operator: ');  Readln (Op);
  if Op = '++' then
    Writeln (N + 1)
  else
    Writeln (N - 1);
end.
```

```
{1.4}
program One4T95;
{ -- This program rounds to three places by break point. }
  var
    BP:  Integer;
    Num: Real;

begin
  Write ('Enter break point: ');  Readln (BP);
  Write ('Enter number: ');  Readln (Num);
  Writeln ( Trunc((Num * 1000 + (10 - BP) / 10)) / 1000 :5:3);
end.


{1.5}
program One5T95;
{ -- This program determines if a program is a REXX or a CLIST. }
  var
    C: String[80];

begin
  Write ('Enter comment: ');  Readln (C);
  if Pos('REXX', C) > 0 then
    Writeln ('REXX')
  else
    Writeln ('CLIST');
end.


{1.6}
program One6T95;
{ -- This program displays the number of times variables appear.}
  var
    Num, Init, Init0: Integer;

begin
  Write ('Enter number of variables: ');  Readln (Num);
  Write ('Enter number initialized: ');   Readln (Init);
  Write ('Enter number initialized to 0: ');  Readln (Init0);
  Writeln ('BASIC = ', Init - Init0);
  Writeln ('PASCAL = ', Num + Init);
  Writeln ('C/C++ = ', Num);
end.
```

```
{1.7}
program One7T95;
{ -- This program displays last qualifier of a data set name. }
  var
    DSN:  String[44];
    Last: String[8];
    I:    Integer;
    Ch:   Char;

begin
  Write ('Enter data set name: ');  Readln (DSN);
  Last := '';
  for I := Length(DSN) downto 1 do begin
    Ch := DSN[I];
    if Ch = '.' then begin
      Writeln (Last); Exit; end
    else
      Last := Ch + Last;
  end;
end.


{1.8}
program One8T95;
{ -- This program displays real numbers in reverse order. }
  var
    I, N: Byte;
    A:    Array[1..10] of String[10];

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter #: ');  Readln (A[I]);
  end;
  Writeln;
  for I := N downto 1 do
    Writeln (A[I]);
end.
```

```
{1.9}
program One9T95;
{ -- This program displays a large X made up of letter X's. }
uses Crt;
  var
    Num, I: Byte;

begin
  Write ('Enter number of X''s: ');  Readln (Num);
  ClrScr;
  for I := 1 to Num do begin
    GotoXY (I, I);  Write ('X');
    GotoXY (Num - I + 1, I);  Write ('X');
  end;
end.


{1.10}
program One10T95;
{ -- This program will display the savings in postage. }
  const
    Cost = 23.33333;
  var
    PS, SS, Oz1, Oz2, Page1, Page2: Integer;

begin
  Write ('Enter # of printed sides: ');  Readln (PS);
  Write ('Enter # of single sided pages: ');  Readln (SS);
  { -- Calculate # of pages and wieght for 1st bill }
  Page1 := PS - 6;  Oz1 := 1;
  Oz1 := Oz1 + (Page1 + 8) div 9;
  { -- Calculate # of pages and wight for 2nd bill }
  Page2 := SS + ((PS - SS + 1) div 2) - 6;
  Oz2 := 1;
  Oz2 := Oz2 + (Page2 + 8) div 9;
  Writeln ((Oz1 - Oz2) * Cost :6:2, ' CENTS SAVED');
end.
```

```
{2.1}
program Two1T95;
{ -- This program finds integral solutions of (X,Y) for AX+BY=C }
var
  A, B, C, X: Integer;
  Y:          Real;

begin
  Write ('Enter A, B, C: ');  Readln (A, B, C);
  X := 1;
  repeat
    Y := (C - A * X) / B;
    if Abs(Y - Trunc(Y)) < 0.001 then begin
      Writeln ('(', X, ',', Y :1:0, ')');  Exit;
    end;
    Inc(X);
  until X > 10000;
end.


{2.2}
program Two2T95;

{ -- This program verifies a number by validating check digit. }
  var
    Part:            String[20];
    Prod, Sum, Code: Integer;
    I, L, Digit, ChkDigit, LastDigit: Byte;

begin
  Write ('Enter part number: ');  Readln (Part);
  L := Length(Part);  Prod := 1;
  for I := 1 to L - 1 do begin
    Val(Copy(Part, I, 1), Digit, Code);
    Sum := Sum + Digit * ((I mod 2) + 1);
  end;
  { -- Subtract units digit of Sum from 9 for check digit }
  ChkDigit := 9 - (Sum mod 10);
  Val(Copy(Part, L, 1), LastDigit, Code);
  if ChkDigit = LastDigit then
    Writeln ('OKAY')
  else
    Writeln ('ERROR -- CHECK DIGIT SHOULD BE ', ChkDigit);
end.
```

```
{2.3}
program Two3T95;
{ -- This program determines # of prizes given of $13 million. }
  var
    Prize: LongInt;
    Pow:   Array[0..7] of LongInt;
    A:     Array[0..6] of Byte;
    I:     Byte;

begin
  Prize := 13000000;
  { -- Same algorithm is used as converting # to base 13 #. }
  Pow[7] := 1;
  for I := 1 to 7 do Pow[7] := Pow[7] * 13;
  for I := 6 downto 0 do begin
    Pow[I] := Pow[I+1] div 13;
    A[I]   := Prize div Pow[I];
    Prize  := Prize mod Pow[I];
  end;
  for I := 0 to 6 do
    Writeln ('$', Pow[I], ' = ', A[I]);
end.
```

```
{2.4}
program Two4T95;
{ -- This program determines the cost of Directory Assistance. }
  var
    DAC, Area:      String[11];
    I, N, LocalDAC: Byte;
    Tot, Cost:      Real;

begin
  Write ('Enter number of DACs: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter DAC: ');  Readln (DAC);
    if DAC = '00' then
      Cost := 3.00
    else if DAC = '1411' then begin
      Inc(LocalDAC);  Cost := 0;  end
    else begin
      Area := Copy(DAC, 2, 3);
      if Area = '813' then
        Cost := 0.25
      else
        if (Area = '305') or (Area = '407') or (Area = '904') then
          Cost := 0.40
        else
          Cost := 0.65;
    end;
    Tot := Tot + Cost;
  end;  { -- for I }
  { -- Every local DAC after the third cost 25 cents }
  if LocalDAC > 3 then
    Tot := Tot + (LocalDAC - 3) * 0.25;
  Writeln (Tot: 5:2, ' DOLLARS');
end.
```

```
{2.5}
program Two5T95;
{ -- This program will display the heading of even/odd pages. }
  const
    PNum: Array [1..4] of Integer = (180, 140, 200, 260);
    P:    Array [1..4] of String[17] =
          ('PROBLEMS', 'JUDGING CRITERIA',
           'BASIC SOLUTIONS', 'PASCAL SOLUTIONS');
  var
    I, Pag, Page, Chapter: Integer;

begin
  Write ('Enter page number: ');  Readln (Page);
  if Page mod 2 = 0 then begin
    Write (Page, '  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION');
    Writeln (' 1985 - 1994');
    end
  else begin
    Write ('FHSCC ''');
    I := 1;  Pag := Page;
    while Pag > PNum[I] do begin
      Pag := Pag - PNum[I];  Inc(I);
    end;
    Chapter := Trunc(Pag / (PNum[I] / 10));
    Writeln (85 + Chapter, ' ', P[I], '  ', Page);
  end;
end.
```

```
{2.6}
program Two6T95;
{ -- This program computes total ESTIMATED PREPARATION TIME. }
  const
    Form: Array[1..6] of String[4] = ('1040','A','B','C','D','E');
    Hr  : Array[1..6,1..4] of Integer = ((3,2,4,0), (2,0,1,0),
          (0,0,0,0), (6,1,2,0), (0,0,1,0), (2,1,1,0));
    Min : Array[1..6,1..4] of Integer = ((8,53,41,53),
          (32,26,10,27), (33,8,17,20), (26,10,5,35),
          (51,42,1,41), (52,7,16,35));
  var
    I, J, TotHr, TotMin: Integer;
    F: String[4];

begin
  I := 0;
  repeat
    Write ('Enter form: ');  Readln (F);
    I := 1;
    while (I < 7) and (F <> Form[I]) do Inc(I);
    if I < 7 then
      for J := 1 to 4 do begin
        Inc(TotHr, Hr[I,J]);
        Inc(TotMin, Min[I,J]);

      end;
  until I > 6;

  Inc(TotHr, TotMin div 60);
  TotMin := TotMin mod 60;
  Writeln (TotHr, ' HR., ', TotMin, ' MIN.');
end.
```

```
{2.7}
program Two7T95;
{ -- This program will calculate investments at GTE. }
  const
    BegPrice: Real = 27.20;
    Return401K: Real = 0.14;
  var
    Salary, Percent, EndPrice, StockGain: Real;
    CompCont, EmpCont, K401, TotalGain:   Real;
    MaxShares, Shares:                    Integer;

begin
  Write ('Enter salary: ');  Readln (Salary);
  Write ('Enter 401K %: ');  Readln (Percent);
  Percent := Percent / 100;
  MaxShares := Trunc(Salary / 100);
  Writeln ('YOU CAN PURCHASE UP TO ', MaxShares, ' SHARES');
  Write ('Enter number of shares: ');  Readln (Shares);
  Write ('Enter end of year price: '); Readln (EndPrice);

  EmpCont := Salary * Percent;
  if Percent >= 0.06 then
    CompCont := (Salary * 0.06) * 0.75
  else
    CompCont := (Salary * Percent) * 0.75;
  K401 := (EmpCont + CompCont) * Return401K;
  StockGain := Shares * (EndPrice - BegPrice);
  TotalGain := CompCont + K401 + StockGain;

  Writeln ('COMPANY CONTRIBUTION: ', CompCont  :8:2);
  Writeln ('         401K RETURN: ', K401      :8:2);
  Writeln ('          STOCK GAIN: ', StockGain :8:2);
  Writeln ('          TOTAL GAIN: ', TotalGain :8:2);
end.
```

```
{2.8}
program Two8T95;
{ -- This program will produce loops of a spiral using letters. }
uses Crt;
  var
    Num, Row, Col, Incr, LoopNum, I: Byte;
    Let: Char;

begin
  Write ('Enter number of spiral loops: ');  Readln (Num);
  Write ('Enter first letter: ');  Readln (Let);
  ClrScr;
  Row := 12;  Col := 40;  Incr := 1;
  while LoopNum < Num do begin
    Incr := Incr + 2;
    { -- Go right }
    GotoXY (Col, Row);  for I := 1 to Incr do Write (Let);
    Col := Col + Incr - 1;
    { -- Go down }
    for I := 1 to Incr - 1 do begin
      GotoXY (Col, Row + I);  Write (Let);
    end;
    Row := Row + Incr - 1;  Incr := Incr + 2;
    { -- Go left }
    Col := Col - Incr + 1;
    GotoXY (Col, Row);  for I := 1 to Incr do Write (Let);
    { -- Go up }
    for I := 1 to Incr - 2 do begin
      GotoXY (Col, Row - I);  Write (Let);
    end;
    Row := Row - Incr + 1;
    if Let = 'Z' then
      Let := 'A'
    else
      Let := Chr(Ord(Let) + 1);
    Inc(LoopNum);
  end;
end.
```

```
{2.9}
program Two9T95;
{ -- This program shows all possible moves for a Queen in chess.}
uses Crt;
  var
    Col, Row, I, J, Code: Integer;
    RC:    String[2];
    R, C:  Array[1..4] of Integer;

begin
  Write ('Enter column and row: ');  Readln (RC);
  Col := Ord(RC[1]) - Ord('A') + 1;
  Val(Copy(RC, 2, 1), Row, Code);
  Row := 9 - Row;
  ClrScr;
  for I := 8 downto 1 do Writeln (I);
  Writeln ('  A B C D E F G H');
  { -- Horizontal moves }
  GotoXY (3, Row);  Writeln ('* * * * * * * *');
  { -- Vertical moves }
  for I := 1 to 8 do begin
    GotoXY (Col * 2 + 1, I);  Write ('*');
  end;
  { -- Diagonal moves }
  for I := 1 to 7 do begin
    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 do
      if (R[J] > 0) and (R[J] < 9) and (C[J] > 0) and (C[J] < 9)
      then begin
        GotoXY (C[J] * 2 + 1, R[J]);  Write ('*');
      end;
  end;
  GotoXY (Col * 2 + 1, Row);  Write('Q');
end.
```

```
{2.10}
program Two10T95;
{ -- This program tabulates information during a pre-election. }
  const
    A: Array[1..10] of String[37] = ('MALE', 'FEMALE',
       '50 AND BELOW', 'OVER 50', 'WHITE', 'OTHERS',
       'ABOVE $25000', '$25000 AND BELOW',
       'WHITE MALE OVER 50 AND ABOVE $25000', 'OTHER');
  var
    Sex, Race, Party:         Char;
    Income:                   LongInt;
    Row, Col, Age, Total:     Byte;
    Sum: Array[1..10,1..2] of Byte;

begin
  Total := 0;
  for Row := 1 to 10 do
    for Col := 1 to 2 do
      Sum[Row, Col] := 0;
  Write ('Enter sex: ');  Readln (Sex);
  while (Sex <> 'E') do begin
    Write ('Enter age: ');    Readln (Age);
    Write ('Enter race: ');   Readln (Race);
    Write ('Enter income: '); Readln (Income);
    Write ('Enter party: ');  Readln (Party);
    if Party = 'D' then Col := 1 else Col := 2;
    if Sex = 'M'   then Row := 1 else Row := 2;
    Inc(Sum[Row,Col]);
    if Age <= 50   then Row := 3 else Row := 4;
    Inc(Sum[Row,Col]);
    if Race = 'W'  then Row := 5 else Row := 6;
    Inc(Sum[Row,Col]);
    if Income > 25000 then Row := 7 else Row := 8;
    Inc(Sum[Row,Col]);
    if (Race = 'W') and (Sex = 'M') and (Age > 50) and (Row = 7)
      then Row := 9 else Row := 10;
    Inc(Sum[Row,Col]);
    Inc(Total);
    Writeln;
    Write ('Enter sex: ');  Readln (Sex);
  end;
  Write (' ':32, 'DEMOCRATIC  REPUBLICAN');
  for Row := 1 to 10 do begin
    if Row mod 2 = 1 then Writeln;
    Write (A[Row], ' ': 37 - Length(A[Row]));
    Write (Sum[Row, 1] / Total * 100 :5:1);
    Writeln (' ':7, Sum[Row,2] / Total * 100 :5:1);
  end;
end.
```

```
{3.1}
program Thr1T95;
{ -- This program will determine how much IRS owes/pays. }
  const
    Amount:    Array[0..5] of Real =
               (0, 22750, 55100, 115000, 250000, 9999999);
    Rate:      Array[0..5] of Real =
               (0, 0.15, 0.28, 0.31, 0.36, 0.396);
    StDeduct:  Real = 3800;
    Exemption: Real = 2450;
  var
    Gross, Deductions, FedTax, Income, TaxInc, Tax: Real;
    I, J: Byte;

begin
    Write ('Enter adjusted gross income: ');  Readln (Gross);
    Write ('Enter itemized deductions: ');  Readln (Deductions);
    Write ('Enter federal income tax withheld: ');
    Readln (FedTax);
    if Deductions > StDeduct then
      Income := Gross - Deductions
    else
      Income := Gross - StDeduct;
    TaxInc := Income - Exemption;

    Tax := 0;
    for I := 1 to 5 do
      if TaxInc <= Amount[I] then begin
        for J := 1 to I - 1 do
          Tax := Tax + (Amount[J] - Amount[J-1]) * Rate[J];
        Tax := Tax + (TaxInc - Amount[I-1]) * Rate[I];
        Write (Abs(Tax - FedTax) :9:2, ' DOLLARS ');
        if FedTax < Tax then
          Writeln ('YOU OWE')
        else
          Writeln ('WILL BE REFUNDED TO YOU');
        Exit;
      end;
end.



{3.2}
program Thr2T95;
{ -- This program will display a simplified phone bill. }
  var
    I, L, HH, Code:          Integer;
    Rate1, Rate2, Tot, Disc: Real;
    Min:     Array[1..10] of Byte;
    Tim:     Array[1..10] of String[13];
    Charge:  Array[1..10] of Real;
    AM, Day: String[3];
    Midday:  Boolean;
```

```
begin
  L := 1;  Tot := 0;
  Write ('Enter MIN: ');  Readln (Min[L]);
  while Min[L] > 0 do begin
    Write ('Enter time: ');  Readln (Tim[L]);
    Inc(L);
    Write ('Enter MIN: ');   Readln (Min[L]);
  end;
  Dec(L);
  { -- Display bill }
  Writeln ('  BOB SMITH  (813) 555-1234');  Writeln;
  Writeln ('  TIME OF DAY  MIN.  CHARGE');
  for I := 1 to L do begin
    if Copy(Tim[I], 1, 1) = '0' then
      Write (' ', Copy (Tim[I], 2, 12))
    else
      Write (Tim[I]);
    { -- Calculate charge }
    Val(Copy(Tim[I], 1, 2), HH, Code);
    AM := Copy(Tim[I], 7, 2);
    Day := Copy(Tim[I], 11, 3);
    Midday := (  (HH > 7)  and (HH < 12) and (AM = 'AM')
              or (HH = 12) and (AM = 'PM')
              or (HH < 5)  and (AM = 'PM') );
    if (HH > 4) and (HH < 11) and (AM = 'PM') and (Day <> 'SAT')
    then
      begin
        Rate1 := 0.21;  Rate2 := 0.16;
      end
    else if Midday and (Day <> 'SAT') and (Day <> 'SUN') then
      begin
        Rate1 := 0.28;  Rate2 := 0.21;
      end
    else
      begin
        Rate1 := 0.14;  Rate2 := 0.11;
      end;
    Charge[I] := Rate1 + Rate2 * (Min[I] - 1);
    Writeln (Min[I] :5, '   ', Charge[I]: 6:2);
    Tot := Tot + Charge[I];
  end;
  if Tot > 20 then Disc := Tot * 0.20;
  Writeln;
  Writeln ('TOTAL CHARGES', ' ': 8,  Tot: 6:2);
  Writeln ('DISCOUNT',      ' ': 13, Disc: 6:2);
  Writeln ('CHARGES - DISCOUNT   ',  Tot - Disc :6:2);
end.
```

```
{3.3}
program Thr3T95;
{ -- This program simulates a baseball game. }
uses Crt;
  var
    I, Inn, T, S, B, W, R, O, Wtot, Otot: Byte;
    Stot, Btot: Integer;
    Run:        Array [1..2] of Byte;

begin
  Randomize;  ClrScr;  Writeln;  Write (' ': 7);
  for I := 1 to 9 do Write (I:3);
  Writeln ('  SCORE');
  Write (' ': 8);
  for I := 1 to 34 do Write ('-');
  Writeln;
  Writeln ('TEAM A !', ' ': 27, '!');
  Writeln ('TEAM B !', ' ': 27, '!');
  Stot := 0;  Btot := 0;  Otot := 0;  Wtot := 0;
  Run[1] := 0;  Run[2] := 0;

  for Inn := 1 to 9 do
    for T := 1 to 2 do begin
      S := 0;  B := 0;  W := 0;  R := 0;  O := 0;
      while O < 3 do begin
        if Random < 0.4 then begin
          Inc(S);  Inc(Stot);  end
        else begin
          Inc(B);  Inc(Btot);
        end;
        if S = 3 then begin
          Inc(O);  Inc(Otot);  S := 0;  W := 0;
        end;
        if B = 4 then begin
          Inc(W);  Inc(Wtot);  B := 0;  S := 0
        end;
        if W = 4 then begin
          Inc(R);  Inc(Run[T]);  W := 3;
        end;
      end;
      GotoXY (6 + Inn * 3, 3 + T);  Write (R:2);
    end;  { -- for T }

  GotoXY (38, 4);  Writeln (Run[1]: 3);
  GotoXY (38, 5);  Writeln (Run[2]: 3);
  Writeln;
  Writeln ('TOTAL # OF STRIKES: ', Stot);
  Writeln ('TOTAL # OF BALLS: ', Btot);
  Writeln ('TOTAL # OF WALKS: ', Wtot);
  Writeln ('TOTAL # OF STRIKE OUTS: ', Otot);
end.
```

```
{3.4}
program Thr4T95;
{ -- This program will produce all possible subsets of letters. }
  var
    Sub:       Array[1..1024] of String[10];
    Let, XSub: String[10];
    A:         Array[1..10] of Char;
    X:         Char;
    I, J, L, Col, SubLen, Bit: Byte;
    N, Num, Two2L, Power:      Integer;

begin
  Write ('Enter letters: ');  Readln (Let);
  L := Length(Let);
  for I := 1 to L do A[I] := Let[I];
  { -- Sort letters in A[] }
  for I := 1 to L - 1 do
    for J := I + 1 to L do
      if A[I] > A[J] then begin
        X := A[I]; A[I] := A[J]; A[J] := X;
      end;
  { -- Generate binary numbers to produce all subsets }
  Two2L := 1;
  for I := 1 to L do Two2L := Two2L * 2;
  for N := 0 to Two2L - 1 do begin
    Num := N; Power := Two2L;  Sub[N] := '';
    for J := L - 1 downto 0 do begin
      Power := Power div 2;
      Bit := Num div Power;
      if Bit = 1 then begin
        Sub[N] := Sub[N] + A[L - J];  Num := Num - Power;
      end;
    end;
  end;
  { -- Bubble sort subsets }
  for I := 0 to Two2L - 2 do
    for J := I + 1 to Two2L - 1 do
      if Sub[I] > Sub[J] then begin
        XSub := Sub[I];  Sub[I] := Sub[J];  Sub[J] := XSub;
      end;
  { -- Display subsets }
  Col := 0;
  for I := 0 to Two2L - 1 do begin
    SubLen := Length(Sub[I]) + 3;
    if Col + SubLen > 50 then begin
      Writeln;  Col := 0;
    end;
    Write ('{', Sub[I], '} ');
    Col := Col + SubLen;
  end;
  Writeln;  Writeln('TOTAL SUBSETS = ', Two2L);
end.
```

```
{3.5}
program Thr5T95;
{ -- This program will sum big integers from 1 to N. }
{ -- Gauss's formula: SUM = N * (N+1) / 2.           }
  var
    A, B, Prod, D:              Array[1..80] of Byte;
    I, J, S, Carry, LenA, LenB: Byte;
    N:                          String[40];
    Code:                       Integer;

begin
  Write ('Enter N: ');  Readln (N);
  { -- Store digits of N in A[] and B[] }
  LenA := Length(N);  LenB := LenA;
  for I := 1 to LenA do begin
    Val(Copy(N, LenA - I + 1, 1), A[I], Code);
    B[I] := A[I];
  end;
  { -- Add 1 to number in B[] }
  Inc(B[1]);  I := 1;
  while (B[I] = 10) do begin
    B[I] := 0;  Inc(I);  Inc(B[I]);
  end;
  if I > LenB then LenB := I;
  { -- Multiply A[] by B[] }
  for I := 1 to LenA do begin
    Carry := 0;
    for J := 1 to LenB do begin
      S := I + J - 1;
      Prod[S] := Prod[S] + A[I] * B[J] + Carry;
      Carry := Prod[S] div 10;
      Prod[S] := Prod[S] - Carry * 10;
    end;
    if Carry > 0 then Prod[S+1] := Carry;
  end;
  if Carry > 0 then Inc(S);
  { -- Divide product Prod[] by 2 }
  if Prod[S] = 1 then begin
    Dec(S);  Carry := 10;
  end;
  for I := S downto 1 do begin
    D[I] := (Prod[I] + Carry) div 2;
    Carry := (Prod[I] mod 2) * 10;
  end;
  { -- Display answer in D[] }
  for I := S downto 1 do  Write (D[I]);
  Writeln;
end.
```

```
{3.6}
program Thr6T95;
{ -- This program will assign values to variables in BASIC code.}
  var
    L, I, PosV, PosV2, PosV3, Num1, Num2, Code: Integer;
    A: Array[1..12] of String[5];
    B: Array[1..12] of Integer;
    V, Ch, Op: Char;
    AllV: String[5];

begin
  L := 0;
  repeat
    Inc(L);
    Write ('Enter line: ');  Readln (A[L]);
  until A[L] = 'END';
  Dec(L);

  AllV := '';
  for I := 1 to L do begin
    { -- Determine if first variable is new or old }
    V := A[I,1];
    PosV := Pos(V, AllV);
    if PosV = 0 then begin
      AllV := AllV + V;
      PosV := Length(AllV);
    end;
    { -- Assign value for first number }
    Ch := A[I,3];
    if (Ch in ['0'..'9']) then
      Val(Ch, Num1, Code)
    else begin
      PosV2 := Pos(Ch, AllV);
      Num1 := B[PosV2];
    end;

    if Length(A[I]) = 3 then
      { -- Assign first number to current variable }
      B[PosV] := Num1
    else begin
      { -- Assign value for second number }
      Ch := A[I,5];
      if Ch in ['0'..'9'] then
        Val(Ch, Num2, Code)
      else begin
        PosV3 := Pos(Ch, AllV);
        Num2 := B[PosV3];
      end;
      { -- Perform operation with 1st and 2nd num, place in var }
      Op := A[I,4];
      Case Op of
        '+': B[PosV] := Num1 + Num2;
        '-': B[PosV] := Num1 - Num2;
        '*': B[PosV] := Num1 * Num2;
```

```
        '/': B[PosV] := Num1 div Num2;
      end;
    end;
  end;  { -- for I }
  { -- Display the variables in order of appearance with values }
  for I := 1 to Length(AllV) do
    Writeln (Copy(AllV, I, 1), '=', B[I]);
end.
```

```
{3.7}
program Thr7T95;
{ -- This program finds three 3-digit primes having digits 1-9. }
  var
    A:           Array[1..200] of LongInt;
    Digits:      String[9];
    Prime, Good: Boolean;
    I, J, K, L, H, T, One, P, Sum, PNum: Integer;
begin      { -- Generate primes into A[] }
  P := 0; I := 101;
  repeat
    J := 3;  Prime := True;
    while (J <= Sqrt(I)) and Prime do begin
      if I mod J = 0 then Prime := False;
      J := J + 2;
    end;
    if prime then begin
      { -- Ensure that Digits are unique and not 0 }
      H := I div 100;  T := (I - H * 100) div 10;
      One := I - H * 100 - T * 10;
      if (T > 0) and (H <> T) and (T <> One) and (H <> One) then
        begin  Inc(P);  A[P] := I;  end;
    end;
    Inc(I, 2);
  until I > 997;
  { -- Add the different combinations of 3 primes }
  for I := 1 to P - 2 do
    for J := I + 1 to P - 1 do
      for K := J + 1 to P do begin
        Sum := A[I] + A[J] + A[K];
        { -- Check if Sum has 4 digits in ascending order }
        if Sum >= 1234 then begin
          Str(Sum, Digits);  Good := True;  L := 1;
          repeat
            if Digits[L] >= Digits[L+1] then Good := False;
            Inc(L);
          until (L = 4) or not Good;
          { -- Check all 3-digit primes for digits 1 through 9 }
          if Good then begin
            Str((((A[I] * 1000 + A[J]) * 1000) + A[K]), Digits);
            L := 1;
            while (L <= 9) and Good do begin
              if Pos(Chr(48+L), Digits) = 0 then Good := False;

              Inc(L);
            end;
            if Good then begin
              Writeln (A[I],' + ',A[J],' + ',A[K],' = ', Sum);
              Inc(PNum);  If PNum = 7 then Exit;
            end;
          end;
        end;
      end;  { -- for K }
end.
```

```
{3.8}
program Thr8T95;
{ -- This program will display time in MM:SS in block letters. }
uses Crt;
  const
    B: Array[1..5] of String[60] = (
    '****     *  ****  ****  *  *  ****  *     ****  ****  ****',
    '*  *     *     *     *  *  *  *     *        *  *  *  *  *',
    '*  *     *  ****  ****  ****  ****  ****     *  ****  ****',
    '*  *     *  *        *     *     *  *  *     *  *  *     *',
    '****     *  ****  ****     *  ****  ****     *  ****     *'
    );
    { -- Maximum units for MM:SS }
    Max: Array[1..4] of Byte = (6, 10, 6, 10);
    { -- Columns to start blocks }
    Col: Array[1..4] of Byte = (1, 7, 18, 24);
  var
    I, J: Byte;
    Dig:  Array[0..9] of Byte;
    A:    Array[1..5,0..9] of String[4];
    MMSS: String[5];
    Code: Integer;
    Ch:   String[1];

begin
  for I := 1 to 5 do
    for J := 0 to 10 do
      A[I,J] := Copy(B[I], J * 6 + 1, 4);
  Write ('Enter MM:SS: '); Readln (MMSS);
  for I := 1 to 4 do
    if I < 3 then
      Val(Copy(MMSS, I, 1), Dig[I], Code)
    else
      Val(Copy(MMSS, I+1, 1), Dig[I], Code);

  ClrScr;
  GotoXY (14,2); Write('*');  GotoXY (14,4);  Write('*');
  Ch := '';
  repeat
    for I := 1 to 4 do
      for J := 1 to 5 do begin
        GotoXY (Col[I], J);  Write (A[J, Dig[I]]);
      end;
    Inc(Dig[4]);
    for J := 4 downto 1 do
      if Dig[J] = Max[J] then begin
        Inc(Dig[J-1]);  Dig[J] := 0;
      end;
    Delay(1000);
    if KeyPressed then Ch := ReadKey;
  until Ch <> ''
end.
```

```
{3.9}
program Thr9T95;
{ -- This program will calculate the area of a polygon room. }
  var
    I, L, Sides, Code, Sum, Area: Integer;
    Mov:  String[3];
    Dir:  Array[1..10] of String[1];
    Dist: Array[1..10] of Integer;

begin
  Write ('Enter number of sides: ');  Readln (Sides);
  for I := 1 to Sides do begin
    Write ('Enter movement: ');  Readln (Mov);
    Dir[I] := Copy(Mov, 1, 1);
    L := Length(Mov);
    Mov := Copy(Mov, 2, L - 1);
    Val(Mov, Dist[I], Code);
    { -- Subtract Down and Left directions }
    if (Dir[I] = 'D') or (Dir[I] = 'L') then
      Dist[I] := -Dist[I];
  end;
  { -- Multiply length by width to obtain rectangle area, }
  { -- then add or subtract area from overall area. }
  I := 1;  Sum := 0;  Area := 0;
  repeat
    Sum := Sum + Dist[I];
    Area := Area + (Sum * Dist[I+1]);
    Inc(I, 2);
  until (I > Sides);
  Writeln ('AREA = ', Abs(Area), ' SQUARE FEET');
end.
```

```
{3.10}
program Thr10T95;
{ -- This program displays versions of libraries on a graph. }
uses Crt;
  var
    Vers, FirstWk, FWkDisp, WkNum, LWkDisp, LastWk, Backup,
    I, Min, Max, TestArea, FirstPreWk, LastPreWk: Integer;

begin
  Write ('Enter version #: ');  Readln (Vers);
  Write ('Enter first week in test: ');  Readln (FirstWk);
  Write ('Enter first week to display, # of weeks: ');
  Readln (FWKDisp, WkNum);
  ClrScr;
  LWkDisp := FWkDisp + WkNum - 1;
  { -- Display week #s at top (units first, then tens) }
  Write (' ': 9);
  for I := FWkDisp to LWkDisp do  Write (I div 10);
  Writeln;  Write (' ': 9);
```

```
  for I := FWkDisp to LWkDisp do  Write (I mod 10);
  Writeln;  Writeln;
  LastWk := FirstWk + 17;
  { -- Compute # of versions to backup from Vers input }
  Backup := (LastWk - FWkDisp) div 6;
  Vers := Vers - Backup;
  FirstWk := FirstWk - 6 * Backup;
  LastWk  := LastWk  - 6 * Backup;
  repeat
    { -- Display Version and indent }
    Write ('R1V');  if Vers < 10 then Write ('0');
    Write(Vers, 'L01 ');
    if FWkDisp <= FirstWk then begin
      Min := FirstWk;
      Write (' ': FirstWk - FWkDisp); end
    else
      Min := FWkDisp;
    if LWkDisp >= LastWk then Max := LastWk else Max := LWkDisp;
    { -- Display TestArea of 1 if Vers even, 2 if odd; P = Prod }
    TestArea := (Vers mod 2) + 1;
    for I := Min to Max do
      if I < FirstWk + 12 then
        Write (TestArea)
      else
        Write ('P');
    Writeln;
    { -- Display Pre-Production Version }
    FirstPreWk := FirstWk + 5;  LastPreWk := FirstWk + 10;
    if (LastPreWk >= FWkDisp) and (FirstPreWk <= LWkDisp) then
      begin
        Write ('R1V');  if Vers - 1 < 10 then Write ('0');
        Write (Vers - 1, 'L88 ');
        if FirstPreWk > FWkDisp then begin
          Min := FirstPreWk;

          Write (' ': FirstPreWk - FWkDisp); end
        else
          Min := FWkDisp;
        if LWkDisp >= LastPreWk then
          Max := LastPreWk
        else
          Max := LWkDisp;
        for I := 1 to Max - Min + 1 do Write ('*');
        Writeln;
      end; { -- if }
    FirstWk := FirstWk + 6;  LastWk := LastWk + 6;
    Inc(Vers);
  until FirstWk > LWkDisp;
end.
```