```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '86 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T86;
{ -- This program will print "THIS IS THE EASIEST PROGRAM!". }
uses Crt;

begin
  ClrScr;
  GotoXY (25, 12);  Writeln ('THIS IS THE EASIEST PROGRAM!');
end.


{1.2}
program One2T86;
{ -- This program will display the sum, difference, and product. }
  var
    Num1, Num2: Integer;

begin
  Write ('Enter two numbers: ');  Readln (Num1, Num2);
  Writeln ('SUM = ', Num1 + Num2);
  Writeln ('DIFFERENCE = ', Num1 - Num2);
  Writeln ('PRODUCT = ', Num1 * Num2);
end.


{1.3}
program One3T86;
{ -- This program will sum 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + ...
  -- until difference between it and the next term is within E. }
  var
    Sum, LastSum, E, Term, Prod: Real;
    I, J:                        Integer;

begin
  Write ('Enter test value E: ');  Readln (E);
  I := 1;
  Sum := 1;  LastSum := 0;
  while (Sum - LastSum) >= E do begin
    Inc(I);
    Term := 1.0 / I;  Prod := 1;
    for J := 1 to I do
      Prod := Prod * Term;
    LastSum := Sum;
    Sum := Sum + Prod;
  end;
  Writeln (LastSum :8:6);
end.
```

```
{1.4}
program One4T86;
{ -- This program will print a check given name and amount. }
uses Crt;
  var
    First, Last, Middle, Init, Amount: String[10];
    I:                                 Integer;

begin
  ClrScr;
  Write ('Enter first name: ');  Readln (First);
  Write ('Enter middle name: '); Readln (Middle);
  Write ('Enter last name: ');   Readln (Last);
  Init := Copy(Middle, 1, 1);
  Write ('Enter amount: ');      Readln (Amount);

  { -- Display border }
  GotoXY (1, 6);
  for I := 1 to 39 do
    Write ('*');
  for I := 1 to 9 do begin
    GotoXY (1, 6+I);  Write ('*');
    GotoXY (39, 6+I); Write ('*');
  end;
  GotoXY (1, 6+10);
  for I := 1 to 39 do
    Write ('*');

  GotoXY (3, 8);  Write ('BEN''S TOWING SERVICE');
  GotoXY (3, 9);  Write ('4563 WRECKER AVENUE');
  GotoXY (3, 10); Write ('WAVERLY, ARKANSAS 45632');
  GotoXY (4, 12); Write ('PAY TO THE ORDER OF ');
  Write (First, ' ', Init, '. ', Last);
  GotoXY (4, 14); Write ('THE SUM OF $', Amount);
  GotoXY (1, 22);
end.
```

```
{1.5}
program One5T86;
{ -- This program will determine which prisoners may be released.}
  var
    Cell: Array [1..100] of 0..1;
    I, J: Integer;

begin
  for I := 1 to 100 do
    Cell[I] := 1;  { -- Initialize all cells open }
  for I := 2 to 100 do begin
    J := 1;
    while J <= 100 do begin
      Cell[J] := 1 - Cell[J];
      Inc(J,I);
    end;
  end;

  for I := 1 to 100 do
    if Cell[I] = 1 then
      Writeln ('CELL ', I);
end.
```

```
{1.6}
program One6T86;
{ -- This program will determine how much money accumulates. }
  var
    Month, Deposit, Rate, Sum: Real;
    Year, J:                   Integer;

begin
  Write ('Enter monthly investment: ');
  Readln (Month);
  Write ('Enter end of year deposit: ');
  Readln (Deposit);
  Write ('Enter annual rate of interest: ');
  Readln (Rate);
  Writeln;
  Rate := Rate / (12*100);  { -- Rate per month in yr in percent }
  Sum := 0;
  for Year := 1 to 20 do begin
    for J := 1 to 12 do begin
      Sum := Sum + Month;
      Sum := Sum + Rate*Sum;
    end;
    Sum := Sum + Deposit;
  end;
  Writeln ('AMOUNT AT END OF YEAR 20 IS $', Sum: 4:2);
end.
```

```
{1.7}
program One7T86;
{ -- This program will drop g in words ending with ing or ings. }
  var
    I, L, LenWord: Integer;
    Sentence:      String[80];
    Word:          String[20];
    End1, End2:    String[4];
    Ch:            Char;

begin
  Write ('Enter sentence: ');  Readln (Sentence);
  Sentence := Sentence + ' ';
  L := Length(Sentence);
  I := 1;  Word := '';
  while I <= L do begin
    Ch := Sentence[I];
    if Ch <> ' ' then
      Word := Word + Ch
    else begin
      LenWord := Length(Word);
      if LenWord >= 4 then begin
        End1 := Copy(Word, LenWord-2, 3);
        End2 := Copy(Word, LenWord-3, 4);
        if End1 = 'ING' then
          Word := Copy(Word, 1, LenWord-1);
        if End2 = 'INGS' then
          Word := Copy(Word, 1, LenWord-2) + 'S';
      end;
      Write (Word, ' ');
      Word := '';
    end;
    Inc(I);
  end;
end.
```

```
{1.8}
program One8T86;
{ -- This program simulates the population growth of rabbits. }
  var
    Init, OverPop: Integer;
    Month, I:      Integer;
    Pop:           Real;
    Dieing:        Boolean;

begin
  Write ('Enter initial population: ');  Readln (Init);
  Write ('Enter point of over population: ');  Readln (OverPop);
  Writeln;
  Pop := Init;
  Dieing := (Pop >= OverPop);
  for Month := 1 to 23 do begin
    If Dieing then
      If (Pop < 2/3 * Init) then
        begin
          Dieing := False;
          Pop := Pop + Pop * 0.2;
        end
      else
        Pop := Pop - Pop * 0.15
    else
      if (Pop >= OverPop) then
        begin
          Dieing := True;
          Init := Trunc(Pop);
          Pop := Pop - Pop * 0.15;
        end
      else
        Pop := Pop + Pop * 0.2;

    Writeln ('POPULATION FOR MONTH ', Month, ' IS ', Pop :2:0);
  end;
end.
```

```
{1.9}
program One9T86;
{ -- This program doubles every e that appears as a single e. }
  var
    Sentence:           String[200];
    LastCh, Ch, NextCh: Char;
    I:                  Integer;
begin
  Write ('Enter sentence: ');  Readln (Sentence);
  I := 1;  LastCh := ' ';
  repeat
    Ch := Sentence[I];
    NextCh := Sentence[I+1];
    if (Ch = 'E') and (LastCh <> 'E') and (NextCh <> 'E') then
      Write ('E');
    Write (Ch);
    Inc(I);
    LastCh := Ch;
  until I = Length(Sentence);
  if (NextCh = 'E') and (LastCh <> 'E') then
    Write ('E');
  Write (NextCh);
end.


{1.10}
program One10T86;
{ -- This program will display common elements of two lists. }
  var
    I, J:    Integer;
    A, B, C: Array [1..12] of Integer;
begin
  for I := 1 to 12 do begin
    Write ('Enter ', I, ' of 12: ');  Readln (A[I]);
  end;
  Writeln;
  for I := 1 to 11 do begin
    Write ('Enter ', I, ' of 11: ');  Readln (B[I]);
  end;

  for I := 1 to 12 do C[I] := 0;
  for I := 1 to 12 do
    for J := 1 to 11 do
      if A[I] = B[J] then C[I] := 1;

  for I := 1 to 12 do
    for J := I + 1 to 12 do
      if (A[I] = A[J]) and (C[J] > 0) then
        Inc(C[J]);

  for I := 1 to 12 do
    if C[I] = 1 then
      Write (A[I], '  ');
end.
```

```
{2.1}
program Two1T86;
{ -- This program will right justify sentence within 65 columns. }
  const
    Col: Integer = 65;
  var
    Sentence, Just:  String[65];
    Word:            Array [1..20] of String[12];
    Ch:              Char;
    I, L, Extra, Ex: Integer;
    WordNum:         Integer;
    TotalCh, SpAve:  Integer;

begin
  Write ('Enter Sentence: ');  Readln (Sentence);
  Sentence := Sentence + ' ';
  L := Length(Sentence);
  I := 1;  WordNum := 1;  Word[WordNum] := '';
  TotalCh := 0;
  { -- Parse Words and calculate Total # of Characters in words }
  while (I <= L) do begin
    Ch := Sentence[I];
    if Ch <> ' ' then
      Word[WordNum] := Word[WordNum] + Ch
    else
      if Word[WordNum] > '' then begin
        TotalCh := TotalCh + Length(Word[WordNum]);
        Inc(WordNum);
        Word[WordNum] := '';
      end;
    Inc(I);
  end;
  Dec(WordNum);

  { -- Display words with SpAve spaces between each one. }
  SpAve := (Col - TotalCh) div (WordNum - 1);
  Extra := (Col - TotalCh) - (SpAve * (WordNum-1));
  for I := 1 to WordNum do begin
    If I <= Extra then Ex := 1
      else Ex := 0;
    Write (Word[I], ' ': SpAve + Ex);
  end;
end.
```

```
{2.2}
program Two2T86;
{ -- This program will produce a repeating pattern with XXX -- }
  var
    X1, X2, D1, D2: String[7];
    TotalXD, Row:   Integer;
    NumX, Rows, I:  Integer;

begin
  Write ('Enter total number of X''s and -''s: ');
  Readln (TotalXD);
  Write ('Enter number of X''s: '); Readln (NumX);
  Write ('Enter number of rows: '); Readln (Rows);

  X1 := '';  X2 := '';  D1 := '';  D2 := '';
  for I := 1 to NumX do begin
    X1 := X1 + 'X';
    D2 := D2 + '-';
  end;
  for I := 1 to TotalXD - NumX do begin
    X2 := X2 + 'X';
    D1 := D1 + '-';
  end;

  for Row := 1 to Rows do begin
    if Row mod 2 = 1 then
      for I := 1 to 4 do
        Write (X1, D1)
    else
      for I := 1 to 4 do
        Write (D2, X2);
    Writeln;
  end;
end.
```

```
{2.3}
program Two3T86;
{ -- This program will code or decode a message. }
  var
    Option, I: Integer;
    St1, St2:  String[27];
    Message:   String[80];
    Ch:        Char;

begin
  St1 := 'ZXCVBNMASDFGHJKLQWERTYUIOP ';
  St2 := 'ABCDEFGHIJKLMNOPQRSTUVWXYZ ';
  repeat
    Writeln;
    Writeln ('1) ENCODE');
    Writeln ('2) DECODE');
    Writeln ('3) END');
    Write   ('Choose: ');  Readln (Option);
    if Option < 3 then begin
      Write ('Enter message: ');  Readln (Message);
      for I := 1 to Length(Message) do begin
        Ch := Message[I];
        if Ch <> ' ' then
          if Option = 1 then  { -- Code message }
            Ch := St1[Ord(Ch) - 64]
          else                { -- Decode message }
            Ch := St2[Pos(Ch, St1)];
        Write (Ch);
      end;
      Writeln;
    end;
  until Option = 3;
end.
```

```
{2.4}
program Two4T86;
{ -- This program finds the unique mode of a set of 15 numbers. }
  var
    A, C:           Array [1..15] of Integer;
    I, J, K, Max:   Integer;
    Mode:           Integer;
    ModeExist:      Boolean;

begin
  for I := 1 to 15 do begin
    Write ('Enter number ', I, ': ');  Readln (A[I]);
  end;

  Max := 1;
  for I := 1 to 14 do begin
    C[I] := 1;
    for J := I + 1 to 15 do
      if A[I] = A[J] then begin
        Inc(C[I]);  { -- Has # of duplicates of elements }
        if C[I] > Max then
          Max := C[I];
      end;
  end;

  { -- Mode exists if only one element occurs Max # of times. }
  ModeExist := False;
  for I := 1 to 14 do
    if (C[I] = Max) then
      if not ModeExist then
        begin
          Mode := A[I];  ModeExist := True;
        end
      else begin
        Writeln ('NO UNIQUE MODE');  Exit;  end;

  if ModeExist then
    Writeln ('MODE IS ', Mode)
  else
    Writeln ('NO UNIQUE MODE');
end.
```

```
{2.5}
program Two5T86;
{ -- This program simulates transactions to a savings accounts. }
  const
    Rate: Real = 0.07;
  var
    Option:                               Integer;
    Balance, Deposit, Withdrawal, Credit: Real;

begin
  Write ('Enter original balance: ');  Readln (Balance);
  Writeln;
  repeat
    Writeln ('1. MAKE A DEPOSIT');
    Writeln ('2. MAKE A WITHDRAWAL');
    Writeln ('3. CREDIT INTEREST');
    Writeln ('4. END');
    Write ('Enter option: ');  Readln (Option);  Writeln;
    case Option of
      1: begin
           Write ('Enter amount to deposit: ');  Readln (Deposit);
           Writeln ('BALANCE BEFORE TRANSACTION $', Balance: 7:2);
           Balance := Balance + Deposit;
           Writeln ('MAKE A DEPOSIT');
         end;
      2: begin
           Write ('Enter amount to withdraw: ');
           Readln (Withdrawal);
           Writeln ('BALANCE BEFORE TRANSACTION $', Balance: 7:2);
           Balance := Balance - Withdrawal;
           Writeln ('MAKE A WITHDRAWAL');
         end;
      3: begin
           Writeln ('BALANCE BEFORE TRANSACTION $', Balance: 7:2);
           Credit := Balance * Rate/12;
           Writeln ('CREDIT INTEREST OF $', Credit: 4:2);
           Balance := Balance + Credit;
         end;
    end;

    if Option < 4 then Write ('NEW ')
      else Write ('FINAL ');
    Writeln ('BALANCE $', Balance: 7:2);
    Writeln;
  until Option = 4;
end.
```

```
{2.6}
program Two6T86;
{ -- This program will sum two positive big numbers. }
  var
    St1, St2:    String[38];
    A, B, C:     Array [1..39] of Integer;
    I, L1, L2,
    MaxL, Carry: Integer;
    Ch:          Char;

begin
  Write ('Enter first number:  ');  Readln (St1);
  Write ('Enter second number: ');  Readln (St2);
  for I := 1 to 39 do begin
    A[I] := 0;  B[I] := 0;
  end;
  L1 := Length(St1);  L2 := Length(St2);
  { -- Put 1st number in A[1..L1], 2nd number in B[1..L2] }
  for I := 1 to L1 do begin
    Ch := St1[ L1-I+1 ];
    A[I] := Ord(Ch) - Ord('0');
  end;
  for I := 1 to L2 do begin
    Ch := St2[ L2-I+1 ];
    B[I] := Ord(Ch) - Ord('0');
  end;

  if L1 > L2 then MaxL := L1
    else MaxL := L2;
  Carry := 0;
  { -- Calculate sum in C[1..MaxL] }
  for I := 1 to MaxL do begin
    C[I] := A[I] + B[I] + Carry;
    if C[I] > 9 then begin
      C[I] := C[I] - 10;
      Carry := 1;
    end
    else Carry := 0;
  end;
  if Carry = 1 then begin
    MaxL := MaxL + 1;
    C[MaxL] := 1;
  end;

  Write ('SUM IS ');
  for I := MaxL downto 1 do
    Write (C[I]);
end.
```

```
{2.7}
program Two7T86;
{ -- This program will perform conversions. }
  const
    Dec: Array [1..6] of String[11] =
      ('INCHES', 'FEET', 'MILES', 'OUNCES', 'POUNDS', 'GALLONS');
    Con: Array [1..6] of Real =
      (2.54, 0.3048, 1.6093, 28.35, 0.4536, 3.7854);
    Met: Array [1..6] of String[11] =
      ('CENTIMETERS', 'METERS', 'KILOMETERS', 'GRAMS',
       'KILOGRAMS', 'LITERS');
  var
    Option, I: Integer;
    X, Y:      Real;
    St:        String[30];

begin
  repeat
    Writeln;
    { -- Display menu options }
    for I := 1 to 6 do begin
      Write (I: 2, ' ');
      if I mod 2 = 1 then
        begin
          St := Met[(I+1) div 2] + ' TO ' + Dec[(I+1) div 2];
          Write (St, ' ': 23 - Length(St));
          Write (I+6: 2, ' ');
          St := Met[(I+7) div 2] + ' TO ' + Dec[(I+7) div 2];
        end
      else
        begin
          St := Dec[I div 2] + ' TO ' + Met[I div 2];
          Write (St, ' ': 23 - Length(St));
          Write (I+6: 2, ' ');
          St := Dec[(I+6) div 2] + ' TO ' + Met[(I+6) div 2];
        end;
      Writeln (St);
    end;
    Writeln ('13 END' :32);
    Write ('Enter option: ');  Readln (Option);

    if Option < 13 then
    if Option mod 2 = 1 then  { -- Convert Metric to English }
      begin
        Write ('Enter number of ', Met[(Option + 1) div 2],': ');
        Readln (X);
        Y := X / Con[(Option + 1) div 2];
        Write ('THIS IS EQUIVALENT TO ', Y:7:3, ' ');
        Writeln (Dec[(Option+1) div 2]);
      end
    else  { -- Convert English to Metric }
      begin
        Write ('Enter number of ', Dec[Option div 2], ': ');
        Readln (X);
        Y := X * Con[Option div 2];
        Write ('THIS IS EQUIVALENT TO ', Y:7:3, ' ');
        Writeln (Met[Option div 2]);
      end;
  until Option = 13;
end.


{2.8}
program Two8T86;
{ -- This program will generate a mortgate amortization. }
uses Crt;
  var
    Rate, Principal, Payment: Real;
    Years, I, C, Month:       Integer;
    YI, TI, MI, MP, OldP:     Real;
    Ch: Char;

function Power({using} X: Real; {raised to the} Y: Integer):
                                                {giving} Real;
{ -- This function simulates the ^ (power) symbol (X to the Y) }
  var
    I: Integer;
    P: Real;
begin
  P := X;
  for I := 1 to Y-1 do
    P := P * X;
  Power := P;
end;

begin
  Write ('Enter principal: ');           Readln (Principal);
  Write ('Enter % rate of interest: ');  Readln (Rate);
  Write ('Enter term in years: ');       Readln (Years);
  Write ('Enter # of month in year for first payment: ');
  Readln (Month);

  Rate := Rate / (12 * 100);
  Payment := (Rate * Power((1+Rate),(Years*12)))/
             (Power((1+Rate),(12*Years)) -1) * Principal;
  C := Month - 1;  OldP := Principal;
  Rate := Rate * 12;  YI := 0;  TI := 0;
  Writeln ('INTEREST         PRINCIPAL');

  for I := 1 to Years*12 do begin
    MI := OldP * Rate/12;
    MP := Payment - MI;
    OldP := OldP - MP;
    Writeln ('$', MI: 6:2, ' ':10, '$', OldP :8:2);
    C := C + 1;  YI := YI + MI;
    if C mod 12 = 0 then begin
      Writeln;
      Writeln ('YEAR''S INTEREST', '  $', YI: 8:2);
      TI := TI + YI;  YI := 0;
      Ch := ReadKey;
    end;
  end;

  if Month <> 1 then begin
    Writeln;
    Writeln ('YEAR''S INTEREST', '  $', YI: 8:2);
    TI := TI + YI;
    Ch := ReadKey;
  end;
  Writeln ('TOTAL INTEREST   $', TI: 8:2);
  Writeln ('MONTHLY PAYMENT  $', Payment: 8:2);
end.


{2.9}
program Two9T86;
{ -- This program calculates the value of sine(x) by a series. }
  var
    N, X, Sum, Factorial, Term: Real;
    I, J, Power:                Integer;

begin
  Write ('Enter N degrees: ');  Readln (N);
  Sum := 0;
  if N > 180 then
    X := Pi * ((360-N)/180)
  else
    X := Pi * (N/180);
  Power := -1;
  for I := 1 to 6 do begin
    Power := Power + 2;
    Factorial := 1;
    for J := 1 to Power do
      Factorial := Factorial * J;
    Term := 1;
    for J := 1 to Power do
      Term := Term * X;
    Term := Term / Factorial;
    if I mod 2 = 1 then
      Sum := Sum + Term
    else
      Sum := Sum - Term;
  end;

  if N > 180 then begin
    Sum := -1 * Sum;  X := Pi * (N/180);
  end;
  Writeln ('PARTIAL SUM = ', Sum :9:7);
  Writeln ('ACTUAL SINE = ', Sin(X) :8:7);
end.
```

```
{2.10}
program Two10T86;
{ -- This program will convert a Roman Numeral to Arabic form. }
  const
    RN: String[7] = 'MDCLXVI';
    RV: Array [1..7] of Integer = (1000, 500, 100, 50, 10, 5, 1);
  var
    RomNum:         String[12];
    I, Ind1, Ind2: Integer;
    L, Arabic:      Integer;
    Ch, NextCh:     Char;

begin
  Write ('Enter Roman Numeral: ');  Readln (RomNum);
  L := Length (RomNum);  I := 1;  Arabic := 0;
  while (I < L) do begin
    Ch     := RomNum[I];     Ind1 := Pos(Ch, RN);
    NextCh := RomNum[I+1];  Ind2 := Pos(NextCh, RN);
    if Ind1 <= Ind2 then { -- value of first is greater or equal}
      Arabic := Arabic + RV[Ind1]
    else begin  { -- value of first is less than second }
      Arabic := Arabic + RV[Ind2] - RV[Ind1];
      Inc(I);
    end;
    Inc(I);
  end;

  if I = L then begin  { -- Last numeral was not done }
    Ch := RomNum[I];  Ind1 := Pos(Ch, RN);
    Arabic := Arabic + RV[Ind1];
  end;
  Writeln ('ARABIC = ', Arabic);
end.
```

```
{3.1}
program Thr1T86;
{ -- This program produces monthly calendars for the year 1986. }
uses Crt;
  const
    Mo: Array[1..12] of String[9] = ('JANUARY','FEBRUARY',
      'MARCH','APRIL','MAY','JUNE','JULY','AUGUST','SEPTEMBER',
      'OCTOBER','NOVEMBER','DECEMBER');
    Days: Array[1..12] of Integer =
      (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    D: Array[1..7] of Char = ('S', 'M', 'T', 'W', 'T', 'F', 'S');

  var
    I, M, Col, Day: Integer;
    Ch: Char;

begin
  ClrScr;
  Writeln (' ':14, '1986');  Writeln;
  for M := 1 to 12 do begin
    { -- Display Month name and Day initials. }
    if M > 1 then ClrScr;
    Writeln (' ':12, Mo[M]);  Writeln;
    for I := 1 to 7 do
      Write (D[I]: 4);
    Writeln;

    { -- Display Day numbers in proper column. }
    if M = 1 then Col := 4;
    if Col > 1 then
      Write (' ': (Col-1)*4);
    for Day := 1 to Days[M] do begin
      Write (Day: 4);
      if Col < 7 then
        Col := Col + 1
      else begin
        Col := 1;  Writeln;
      end;
    end;
    Ch := ReadKey;
  end;
end.
```

```
{3.2}
program Thr2T86;
{ -- This program finds the root of a 5th degree polynomial }
{ -- of the form Ax^5 + Bx^4 + Cx^3 + Dx^2 + Ex + F = 0.    }
  var
    A, B, C, D, E, F: Real;
    X, X1, X2:  Real;

function Y(X,A,B,C,D,E,F: Real):Real;
{ -- This function returns value of Y given coefficients and X. }
begin
  Y := A*X*X*X*X*X + B*X*X*X*X + C*X*X*X + D*X*X + E*X + F;
end;

begin
  Write ('Enter coefficients A,B,C,D,E,F: ');
  Readln (A,B,C,D,E,F);
  { -- This algorithm finds 1 and only 1 root (closest to x=0) }
  X1 := -1.0;  X2 := 1.0;
  { -- Find sign change between X1 and X2. }
  while Y(X1,A,B,C,D,E,F) * Y(X2,A,B,C,D,E,F) > 0 do begin
    X1 := X1 - 1;  X2 := X2 + 1;
  end;
  { -- Use binary search to find root. }
  while X2 - X1 > 0.000005 do begin
    X := (X1 + X2) / 2;
    if Y(X,A,B,C,D,E,F) * Y(X1,A,B,C,D,E,F) > 0 then X1 := X
      else X2 := X;
  end;
  Writeln ('ROOT = ', X: 7:5);
end.
```

```
{3.3}
program Thr3T86;
{ -- This program changes a number from one base to another. }
  const
    D: String[36] = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
  var
    A, B, I, J, Ex, X: Integer;
    N, Pow:            Real;
    NumSt:             String[10];

begin
  Write ('Enter base A: ');   Readln (A);
  Write ('Enter base B: ');   Readln (B);
  Write ('Enter original number: '); Readln (NumSt);
  Writeln;  Write (NumSt, ' BASE ', A, ' EQUALS ');
  { -- Convert Num to Base 10 from base A. }
  N := 0;
  for I := 1 to Length(NumSt) do begin
    Pow := 1;
    for J := 1 to Length(NumSt)-I do Pow := Pow * A;
    N := N + (Pos(Copy(NumSt,I,1),D) - 1) * Pow;
  end;

  Ex := 0;  Pow := 1;
  while Pow <= N do begin
    Inc(Ex);  Pow := Pow * B;
  end;
  Dec(Ex);

  { -- Convert Num to Base B from Base 10. }
  for I := Ex downto 0 do begin
    Pow := Pow / B;
    X := Trunc(N / Pow + 0.01);
    Write (D[X+1]);
    N := N - X*Pow;
  end;
  Write (' BASE ', B);
end.
```

```
{3.4}
program Thr4T86;
{ -- This program will update customers account by SSN's. }
  var
    SS:    Array[1..6] of String[9];
    N:     Array[1..6] of String[12];
    A:     Array[1..6] of String[41];
    B:     Array[1..6] of Real;
    SSN:   String[10];
    Temp:  String[41];
    I,J,L: Integer;
    Ch:    Char;
    Trans: Real;
    P1,P2: Integer;

begin
  SS[1] := '234567890'; N[1] := 'JOHN SMITH  ';
  SS[2] := '564783219'; N[2] := 'GAIL HUSTON ';
  SS[3] := '873421765'; N[3] := 'TIM JONES   ';
  SS[4] := '543876543'; N[4] := 'JILL RUPERTS';
  SS[5] := '345212342'; N[5] := 'AL BROWN    ';
  SS[6] := '565656565'; N[6] := 'KERMIT TEU  ';
  A[1]  := '1234 ANYWHERE LANE, EXIST, KANSAS 66754  ';
  A[2]  := '543 SOUTH THIRD, BIG TOWN, TEXAS 88642   ';
  A[3]  := '2387 PALM PLACE, NOME, ALASKA 77643      ';
  A[4]  := '4536 123RD STREET, TINY TOWN, MAINE 76765';
  A[5]  := 'PO BOX 234, TINSEL TOWN, CALIFORNIA 77654';
  A[6]  := '1234 LOST LANE, WIMPLE, WISCONSIN 66543  ';
  B[1]  :=  345.78;
  B[2]  := 2365.89;
  B[3]  := 6754.76;
  B[4]  :=   45.18;
  B[5]  := 3456.09;
  B[6]  :=   78.36;

  Write ('Enter SSN: ');  Readln (SSN);
  while SSN <> '000000000' do begin
    I := 1;
    while (SS[I] <> SSN) and (I < 6) do I := I + 1;
    Write ('Enter C for Charge or P for Payment: ');  Readln(Ch);
    Write ('Enter amount of transaction: ');  Readln(Trans);
    if Ch = 'C' then
      B[I] := B[I] - Trans
    else
      B[I] := B[I] + Trans;
    Writeln;
    Writeln ('NEW BALANCE IS $', B[I]: 5:2);
    Writeln;
    Write ('Enter SSN: ');  Readln (SSN);
  end;
  { -- Sort customers in decreasing order according to balance. }
  for I := 1 to 5 do
    for J := I + 1 to 6 do
      if B[I] < B[J] then begin
        Temp := SS[I]; SS[I] := SS[J];  SS[J] := Temp;
        Temp := N[I];   N[I] := N[J];    N[J] := Temp;
        Temp := A[I];   A[I] := A[J];    A[J] := Temp;
        Trans := B[I];  B[I] := B[J];    B[J] := Trans;
      end;
  { -- Display report }
  Writeln;
  Write   ('SSN', ' ':8, 'NAME', ' ': 10, 'ADDRESS', ' ':2);
  Writeln ('BALANCE': 18);  Writeln;
  for I := 1 to 6 do begin
    Temp := SS[I] + '  ' + N[I] + '  ';
    Write (Temp);
    L := Length(Temp) - 1;
    P1 := Pos(',', A[I]);  Delete(A[I], P1, 1);
    P2 := Pos(',', A[I]);
    Write (Copy(A[I], 1, P1 - 1));
    Writeln ('$': 22 - P1, B[I]:7:2);
    Writeln (' ': L, Copy(A[I], P1, P2 - P1));
    Writeln (' ': L, Copy(A[I], P2+1, Length(A[I]) - P2 - 1));
  end;
  Writeln;
end.
```

```
{3.5}
program Thr5T86;
{ -- This program will print the product of 2 large decimals. }
  var
    AStr, BStr:                       String[31];
    LenA, LenB, ADec, BDec, RDigits: Integer;
    A, B, Prod:                       Array[1..61] of Integer;
    I, J, S, Carry, Base:             Integer;
    Sign: -1..1;

begin
  Write ('Enter first number: ');   Readln (AStr);
  Write ('Enter second number: ');  Readln (BStr);

  { -- Determine # of Digits to the right of decimal in product }
  ADec := Pos ('.', AStr);  BDec := Pos ('.', BStr);
  Delete (AStr, ADec, 1);   Delete (BStr, BDec, 1);
  LenA := Length(AStr);     LenB := Length(BStr);
  RDigits := LenA - ADec + LenB - BDec + 2;

  { -- Store String digits into numerical arrays. }
  for I := LenA downto 1 do
    A[LenA-I+1] := Ord(AStr[I]) - 48;
  for I := LenB downto 1 do
    B[LenB-I+1] := Ord(BStr[I]) - 48;
  for I := 1 to 61 do Prod[I] := 0;

  { -- Multiply 2 numbers as a person would with carries. }
  for I := 1 to LenB do begin
    Carry := 0;
    for J := 1 to LenA do begin
      S := I + J - 1;
      Prod[S] := Prod[S] + B[I]*A[J] + Carry;
      Carry := Prod[S] div 10;
      Prod[S] := Prod[S] - Carry*10;
    end;
    If Carry > 0 then Prod[S+1] := Carry;
  end;

  { -- Display digits of product before decimal }
  Write ('PRODUCT = ');
  if Carry > 0 then Inc(S);
  if S > RDigits then
    for I := S downto RDigits+1 do
      Write (Prod[I])
   else
     Write ('0');
  Write ('.');
  { -- Display digits after decimal. }
  for I := RDigits downto 1 do
    Write (Prod[I]);
end.
```

```
{3.6}
program Thr6T86;
{ -- This program will determine if a # can become palindrome. }
  var
    B, Rev:           Array[1..50] of Integer;
    I, L, Try, Carry: Integer;
    Pal:              Boolean;
    NumSt:            String[10];

begin
  Write ('Enter number: ');  Readln (NumSt);
  L := Length(NumSt);
  for I := 1 to L do
    B[L-I+1] := Ord(NumSt[I]) - 48;
  Try := 0;

  repeat
    { -- Test for Palindrome }
    Pal := True;
    for I := 1 to (L div 2) do
      if B[I] <> B[L-I+1] then Pal := False;

    { -- Add reverse of number to itself. }
    if not Pal then begin
      for I := 1 to L do Rev[I] := B[L-I+1];
      Carry := 0;
      for I := 1 to L do begin
        B[I] := B[I] + Rev[I] + Carry;
        Carry := B[I] div 10;
        B[I] := B[I] - Carry*10;
      end;
      if Carry = 1 then begin
        Inc(L);  B[L] := 1;
      end;
      Inc(Try);
    end;
  until Pal or (Try > 23);

  { -- Display # if Palindrome else say it is not. }
  if Pal then begin
    for I := L downto 1 do Write (B[I]);
    Writeln (' IS A PALINDROME');
    end
  else
    Writeln ('CANNOT GENERATE A PALINDROME');
end.
```

```
{3.7}
program Thr7T86;
{ -- This program will solve an N x N system of equations. }
  var
    C:              Array[1..5,1..6] of Real;
    N, Row, Col, R: Integer;
    Den, X:         Real;

begin
  { -- Enter values in C array }
  Write ('Enter N: ');  Readln (N);
  for Row := 1 to N do begin
    Writeln ('Enter coefficients for Row ', Row);
    for Col := 1 to N do begin
      Write ('Co', Col, ': ');
      Readln (C[Row,Col]);
    end;
    Write ('Enter constant: ');  Readln (C[Row, N+1]);
  end;

  { -- Make main diagonals all 1s with 0s to the left. }
  for Row := 1 to N do begin
    Den := C[Row, Row];
    for Col := Row to N+1 do
      C[Row, Col] := C[Row, Col] / Den;
    for R := Row+1 to N do begin
      X := C[R, Row];
      for Col := Row to N+1 do
        C[R,Col] := C[R,Col] - X * C[Row,Col];
    end;
  end;

  { -- Make 0s on right of 1s on main diagonal, (not constants).}
  for Row := N downto 1 do
    for R := Row-1 downto 1 do begin
      X := C[R, Row];
      for Col := Row to N+1 do
        C[R,Col] := C[R,Col] - X * C[Row,Col];
    end;

  { -- Display solution }
  Write ('(', C[1,N+1]: 1:0);
  for Row := 2 to N do
    Write (', ', C[Row,N+1]: 1:0);
  Writeln (')');
end.
```

```
{3.8}
program Thr8T86;
{ -- This program prints Kth, 2*Kth, and 3*Kth permutations. }
  var
    F, I, J, K,
    L, KK, T, X, S: Integer;
    AStr:           String[7];
    A:              Array[1..7] of Char;
    B:              Array[1..7] of 0..1;
    Temp:           Char;
    Fact:           Array[1..7] of Integer;
    Quit:           Boolean;
begin
  Write ('Enter word: ');  Readln (AStr);
  Write ('Enter K: ');     Readln (K);
  L := Length (AStr);
  { -- Store and alphabetize letters. }
  for I := 1 to L do A[I] := AStr[I];
  for I := 1 to L-1 do
    for J := I+1 to L do
      if A[I] > A[J] then begin
        Temp := A[I];  A[I] := A[J];  A[J] := Temp;
      end;

  { -- Compute Factorials F[3] = 2!, F[4] = 3!... }
  for I := 1 to L do begin
    F := 1;
    for J := 1 to I-1 do F := F * J;
    Fact[I] := F;
  end;

  { -- Generate permutations in order. }
  for T := 1 to 3 do begin
    KK := K*T-1;
    for I := 1 to 7 do B[I] := 0;
    for I := L downto 1 do begin
      X := KK div Fact[I];  S := 0;
      J := 1;  Quit := False;
      repeat
        if B[J] = 0 then begin
          Inc(S);
          if S > X then begin
            B[J] := 1;
            Write (A[J]);
            Quit := True;
          end;
        end;
        Inc(J);
      until (J > L) or Quit;
      KK := KK - Fact[I]*X;
    end;  { -- for I }
    Write('  ');
  end;  { -- for T }
end.
```

```
{3.9}
program Thr9T86;
{ -- This program will solve cryptarithm puzzle ABB - CB = DEF. }
{ -- F = 0 since B-B=0.  A=D+1 or A=D since CB is 2 digits,
     but A<>D. D>B, otherwise D=A. Since B<C, B<9, => E=10+B-C. }
  var
    A, B, C, D, E, F, Tot: Integer;

begin
  Tot := 0;
  for B := 1 to 8 do
    for C := B+1 to 9 do
      for D := 1 to 8 do begin
        F := 0;
        A := D + 1;
        E := 10 + B - C;
        if not ((A=B) or (A=C) or (A=D) or (A=E) or (A=F) or
                (B=C) or (B=D) or (B=E) or (B=F) or (C=D) or
                (C=E) or (C=F) or (D=E) or (D=F)) then begin
           Tot := Tot + 1;
           Writeln (A,B,B,' - ',C,B,' = ',D,E,F,'  NUMBER ',Tot);
         end;
      end;  { -- for D }
  Writeln;
  Writeln ('TOTAL NUMBER OF SOLUTIONS = ',Tot);
end.
```

```
{3.10}
program Thr10T86;
{ -- This program will find all 2-digit integers equal to the sum
     of integers in which each digit 0-9 is used exactly once. }
{ -- Array D is array of digits to appear in Ten's position.
  -- C is count of how many digits are in array D.
  -- S is sum of digits not in array D
  -- F is flag array showing which digits are not in array D. }

  var
    I, J, K, C, DD, N, S, D1, D2, D3, P: Integer;
    F, D: Array[0..9] of Integer;

procedure CheckCondition;
{ -- This procedure will Check the condition. }
begin
  S := 0;  F[0] := 1;
  for I := 1 to 9 do F[I] := 0;
  for I := 1 to 9 do
    if not ((C=1) and (I=D1) or (C=2) and ((I=D1) or (I=D2)) or
           (C=3) and ((I=D1) or (I=D2) or (I=D3))) then begin
      S := S + I;  F[I] := 1;
    end;
  if C = 1 then DD := D1;
  if C = 2 then DD := D1 + D2;
  if C = 3 then DD := D1 + D2 + D3;
  if DD * 10 + S = N then begin
    Write (N, ' = ');
    K := 0;
    for J := 1 to C do begin
      while F[K] = 0 do K := K + 1;
      Write (D[J], K, ' + ');
      Inc(K);
    end;
    for I := K to 9 do begin
      if F[I] = 1 then begin
        Write (I);
        if I < 9 then Write (' + ');
      end;
    end;
    Writeln;
    P := 1;
  end;
end;

begin
  for N := 45 to 99 do begin
    for D1 := 1 to 2 do begin
      D[1] := D1;
      for D2 := D1+1 to 3 do begin
        D[2] := D2;
        for D3 := D2+1 to 4 do begin
          D[3] := D3;  C := 3;  CheckCondition;
        end;
      end;
    end;  { -- for D1}
    D3 := 0;
    if P <> 1 then begin
      for D1 := 1 to 2 do begin
        D[1] := D1;
        for D2 := D1+1 to 3 do begin
          D[2] := D2;  C := 2;  CheckCondition;
        end;
      end;
      D2 := 0;
      if P <> 1 then begin
        for D1 := 1 to 6 do begin
          D[1] := D1;  C := 1;  CheckCondition;
        end;
        if N = 45 then begin
          Write (N, ' = ');
          K := 0;
          for I := K to 9 do begin
            if F[I] = 1 then begin
              Write (I);
              if I < 9 then Write (' + ');
            end;
          end;
          Writeln;
          P := 1;
        end;
      end;  { -- if P<>1 }
    end;  { -- if P<>1 }
    P := 0;
  end;  { -- for N }
end.
```