Supporting Information for:

Nelson W. Green;[1,2] E. Michael Perdue.[2] Fast Graphically Inspired Algorithm for Assignment of Molecular Formulae in Ultrahigh Resolution Mass Spectrometry. *Anal. Chem.* **2015**, 87, 10.1021/ac504166t.

[1]Chemical and Biomolecular Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States

[2]Department of Chemistry, Ball State University, Muncie, Indiana 47306, United States

Email: emperdue@bsu.edu. Phone: +001765858096

Example of CHOFIT program S-2

In the following example, the CHOFIT [*n*] and CHOFIT [*n-3*] algorithms, as shown in Figure 1 and Figure 2, are iterated with an arbitrary mass (formula) and select iterative steps are shown. Real values are given to the user-defined inputs in the above figures, and real masses (sums) for the iterative steps are calculated. The error term,$(EM - EM_{calc})^2 \leq tolerance$, is redefined in terms of ppm error, $10^6\left(\frac{\sqrt{(EM-EM_{calc})^2}}{EM}\right) \leq tolerance$. Iterative steps were counted for each pass through Figure 1, including **ENSP** combinations that cause EM_{core} to be odd, and for each pass through Figure 2. The maximum number of iterative steps in Figure 2d is dependent on the core mass from Figure 1 by the i_{max} equation. Note that the iterative steps in different algorithms do not necessarily take the same time to execute, and because of that a ratio of iterative steps is only an approximation of the ratio of rates. Using this approximation, the rate of CHOFIT [n-3] is expected to be much faster than CHOFIT [n] because CHOFIT [*n*] took many more iterative steps.

CHOFIT [*n-3*] program source code S-3 to S-17

CHOFIT [n-3] program source code in text form is provided below. After saving the text in a Pascal file (CHOFIT3_MIN.pas), it can be compiled with the FreePascal compiler (http://www.freepascal.com/download/). This program can read in limited user inputs from the command line, read a mass list from a text file, and output the results to a text file. The command line format is CHOFIT3_MIN InputFileName OutputFileName Low_MW High_MW N S P 13C (where the element values are the upper limits). An example command line is "CHOFIT_MIN <inputfilename> outputfilename 150 1500 10 6 4 1", where <inputfilename> is replaced by the input (mass list) file name. The default mass lists are text files (.dat) with a list of ion exact masses (IEM). IEM for the four mass lists tested are in **Supporting Information**. The default output is a text file (.fit).

Example of CHOFIT programs (Figure 1 and Figure 2)

Molecular Formulae (Mass)	$^{13}CC_{20}H_{45}O_{10}N_{2}S_{3}P$ (613.200750 Da)				
User Inputs	EM = 613.200750 Da	Tolerance = 0.4 ppm	$C_{max} = 83$	$H_{max} = 144$	$O_{max} = 36$
	$E_{max} = 1$	$N_{max} = 10$	$S_{max} = 6$	$P_{max} = 4$	
Initial values	$n_C = 1$	$n_H = 0$	$n_O = 0$		
	$n_E = 0$	$n_N = 0$	$n_S = 0$	$n_P = 0$	

CHOFIT [*n*] algorithm						
Iteration	**ENSP** Formula	$\sum n_i EM_i$ i=**E**,**N**,S,**P**	CHO Formula	$\sum n_i EM_i$ i=C,H,O	$\sum n_i EM_i$ i=**E**,C,H,O,**N**,**P**,S	Error (ppm)
1		0.000000	C	12.000000	12.000000	980430.55
10584		0.000000	$C_{28}H_{52}O_{14}$	612.335706	612.335706	1410.70
1352136	$\mathbf{EN_2}$	31.025153	$C_{23}H_{34}O_{17}$	582.179600	613.204753	6.53
80033331	$\mathbf{EN_2S_3P}$	158.922953	C	12.000000	170.922953	721261.00
80038171	$\mathbf{EN_2S_3P}$	158.922953	$C_{20}H_{42}O_{10}$	442.277798	613.200750	0.00
The total iterative steps are 770 x 224,183 = 172,620,910. Fewer iterative steps are taken when C_{max}, H_{max}, and O_{max} are constrained by mass.						

CHOFIT [*n-3*] algorithm						
Iteration	**ENSP** Formula	$\sum n_i EM_i$ i=**E**,**N**,S,**P**	CHO Formula	$\sum n_i EM_i$ i=C,H,O	$\sum n_i EM_i$ i=**E**,C,H,O,**N**,**P**,S	Error (ppm)
1	**E**	1.003355	$C_{50}H_{12}$	612.093900	613.097255	168.78
4	**E**	1.003355	$C_{38}H_{12}O_{9}$	612.048132	613.051487	243.42
1851	$\mathbf{EN_2S_3P}$	158.941984	$C_{37}H_{10}$	454.078250	613.	294.38
1857	$\mathbf{EN_2S_3P}$	158.941984	$C_{21}H_{42}O_{10}$	454.258766	613.200750	0.00
The total iterative steps are 3,572.						

CHOFIT[*n-3*] program source code

```
PROGRAM CHOFIT3_MIN;

{=====================================================================
AUTHORS:  Nelson W. Green (Georgia Institute of Technology)
          E. Michael Perdue (Ball State University)

DATE   :  March 15, 2015

CHOFIT3_MIN.PAS is a simplified version of CHOFIT3.PAS, in which it is
assumed that all ions have a charge of -1 due to loss of H(+) ion.  As
presented, CHOFIT3_MIN.PAS does not use Na to fit a molecular formula to
an exact mass.  The program is provided as Supporting Information for
the following paper:

Nelson W. Green and E. Michael Perdue (2015) "Fast graphically-inspired
algorithm for assignment of molecular formulae in ultrahigh resolution
mass spectrometry", Anal. Chem., submitted for publication.

The source code can be compiled using the Free Pascal compiler, which
is available online at www.freepascal.org.  If the required command-line
parameters are provided under the "Run|Parameters..." menu, then the
program may be executed from within the Free Pascal IDE, .  Alternatively,
the standalone executable program that is generated by the compiler may be
executed from a command prompt.  The user must copy the executable program
(CHOFIT3_MIN.EXE) and any data files to a folder and then open a command
window and navigate to that folder.  The program is launched from a
command prompt using the following syntax:

    CHOFIT3_MIN InputFileName OutputFileName Low_MW High_MW N S P 13C

where InputFileName and OutputFileName may be entered with/without
extensions.  If a file extension is omitted, then a default extension
of ".dat" will be appended to InputFileName and a default extension of
".fit" will be appended to OutputFileName.

Low_MW and High_MW are the limits of nominal mass that will be evaluated.

N, S, P, and 13C are the upper limits for the number of these atoms that
can be used in a molecular formula.  The maximum values are N=24, S=8,
P=4, and 13C=1. Lower limits for these elements/isotopes are all zero.
=====================================================================}

{=====================================================================
Global type declarations, constants, and variables for CHOFIT3_MIN.PAS
=====================================================================}

CONST
  Version = 'CHOFIT3_MIN 20150315 Test Version'; {Current version    }

TYPE
  PathStr    = STRING;
  Component  = (C,H,O,N,S,P,M,E,Z);
  IonType    = (Negative,Positive);
  FitType    = (mDa,ppm);
  IFormula   = ARRAY[Component] OF WORD;
  RFormula   = ARRAY[Component] OF REAL;

  Data = RECORD
           ID       : LONGINT;         {ID of mass peak              }
           CNM      : WORD;            {Calculated nominal mass      }
           Moles    : IFormula;        {Molecular formula            }
           IEM      : REAL;            {Exact mass of ñ1 ion         }
           XEM      : REAL;            {Exact mass of molecule       }
           CEM      : REAL;            {Calculated exact mass        }
           Intensity: REAL;            {Peak intensity - not used    }
           XEMerr   : REAL;            {Mass fitting error, mDa/ppm  }
           Fit      : BOOLEAN;         {TRUE if found}
         END;
  FitPtr = ^FitRec;                    {Pointer type for fit records }
  FitRec = RECORD                      {Record type for linked data  }
             Prev: FitPtr;
             Peak: Data;
             Next: FitPtr;
           END;

CONST
  Proton  = 1.00727645216;             {Exact mass of H+        }
  MinMW   : WORD = 0;                  {Program minimum MW      }
  MaxMW   : WORD = 2000;               {Program maximum MW      }
  LowMW   : WORD = 150;                {User minimum MW         }
  HighMW  : WORD = 1000;               {User maximum MW         }
  Digits  : BYTE = 6;                  {Number of decimal places}
  MinFit  : REAL = 0;                  {Program minimum error   }
  MaxFit  : REAL = 2;                  {Program maximum error   }
  MaxErr  : REAL = 0.40;               {Maximum fitting error   }
  FitMode : FitType = ppm;             {Error units - ppm, mDa  }
  IonMode : IonType = Negative;        {Negative/Positive ions  }
  TotalPeaks: LONGWORD = 0;            {Total no. of mass peaks }
  TotalFormulae: LONGWORD =0;          {Total molecular formulae}

{=====================================================================
Nominal masses of components, using zero for charge
=====================================================================}

  NM: IFormula =
    ( 12,            1,            16,              {C        H      O}
      15,           32,            32,              {HN       S     HP}
      22,            1,             0);             {Na-H  13C-12C   Z}

{=====================================================================
Exact masses of components, using the mass of an electron for charge
=====================================================================}

  EM: RFormula =                                    {IUPAC 2003 Masses}
    ( 12.0000000000,  1.0078250319, 15.9949146223,  {C        H      O}
      15.0108990393, 31.9720707300, 31.9815865219,  {HN       S     HP}
      21.9819446281,  1.0033548380,  0.0005485799); {Na-H  13C-12C   Z}

{=====================================================================
The program limits on minimum and maximum moles are:
```

```
======================================================================}

                   {  C   H   O   N   S   P   M   E   Z}
  Min : IFormula = (  1,  2,  0,  0,  0,  0,  0,  0,  1);
  Max : IFormula = (166,284, 72, 24,  8,  4,  2,  1,  5);

{======================================================================
User-specified limits on minimum and maximum moles are:
======================================================================}

                   {  C   H   O   N   S   P   M   E   Z}
  Low : IFormula = (  1,  2,  0,  0,  0,  0,  0,  0,  1);
  High: IFormula = (  1,  2,  0,  0,  0,  0,  0,  0,  1);   {****}

{======================================================================
Valences of the Components:
======================================================================}

                       {C, H, O, N, S, P, M, E, Z}
  Valence: IFormula =  (4, 1, 2, 2, 2, 2, 0, 0, 0);

VAR
  InFile,                             {Name of input data file     }
  InPath,                             {File path for data file     }
  OutFile,                            {Name of output file         }
  OutPath   : PathStr;                {File path for output file   }
  DevI,                               {Input file device           }
  DevO      : TEXT;                   {Output file device          }
  Peak      : Data;                   {The master variable for mass}
  Finished  : BOOLEAN;                {TRUE if program is closed   }
  BaseRec,                            {Pointer to first fit record }
  LastRec,                            {Pointer to last fit record  }
  PeakRec   : FitPtr;                 {Pointer to any fit record   }

{======================================================================
The RoundTo function rounds the real number X to the specified number
of decimal places.
======================================================================}

FUNCTION RoundTo (X: REAL; Places: BYTE): REAL;
BEGIN
  RoundTo:=ROUND(X*EXP(Places*LN(10)))/(EXP(Places*LN(10)));
END;

{======================================================================
The Exists function checks if the file path is valid
======================================================================}

FUNCTION Exists (FileSpec: PathStr): BOOLEAN;
  VAR Dev: FILE;
  BEGIN
    {$I-}
    ASSIGN (Dev,FileSpec);
    RESET (Dev);
    CLOSE (Dev);
    {$I+}
    Exists:=(IORESULT=0) AND (FileSpec<>'');
END;

{=====================================================================
BuildFitRec creates a doubly linked list to store each mass and the
molecular formulae that are assigned to that mass.  BaseRec is the
first record in the list of peaks and solutions for those peaks.
PeakRec is the first solution for the current Peak.  BuildFitRec is
called once for each new molecular formula, so TotalFormulae is
incremented here.
=====================================================================}

PROCEDURE BuildFitRec (Peak: Data; Prev,Next: FitPtr; VAR M: FitPtr);
BEGIN
  NEW(M);
  M^.Peak:=Peak;
  M^.Prev:=Prev;
  M^.Next:=Next;
  IF Prev=Next THEN
  BEGIN
    BaseRec:=M;
    PeakRec:=M;
  END;
  IF Prev<>NIL THEN
  BEGIN
    Prev^.Next:=M;
    IF Prev^.Peak.ID <> M^.Peak.ID THEN PeakRec:=M;
  END;
  IF Next<>NIL THEN Next^.Prev:=M;
  TotalFormulae:=SUCC(TotalFormulae);
END;

{=====================================================================
DeleteFitRecs deletes a doubly linked list of molecular formulae.
=====================================================================}

PROCEDURE DeleteFitRecs (VAR Target: FitPtr);
VAR P,N: FitPtr;
BEGIN
  WHILE Target <> NIL DO
  BEGIN
    P:=Target;
    N:=Target^.Next;
    DISPOSE(P);
    P:=NIL;
    Target:=N;
  END;
END;

{=====================================================================
The function Valid evaluates a molecular formula to ensure that its
composition obeys the Senior Rules and meets other compositional
constraints that X >= Low[X] for X=CHONSPME and O <= (C+2+3*N+4*S+4*P).
=====================================================================}

FUNCTION Valid (Moles: IFormula): BOOLEAN;
```

```
VAR
  OK : BOOLEAN;
  Sum: INTEGER;
  I  : Component;

BEGIN
  {
  ----------------------------------------
  Minimum constraints on CHONSPME formulae
  ----------------------------------------
  }
  OK:=TRUE;
  FOR I:=C TO E DO OK:=OK AND (Moles[I]>=Low[I]);
  {
  The Senior Rules rely heavily on valence, which is an elemental property.
  CHOFIT uses components rather than elements, so the valences used in CHOFIT
  are those of components (e.g., NH has a valence of 2), and they work for
  most tests.  Components that are exchange operators (M and E) are not
  included in these tests because they do not alter the number of bonds in a
  molecule.

  ------------------------------------
  Senior Rule #1 for CHONSPME formulae
  ------------------------------------
    The sum of atoms having odd valences must be an even number.  The use of
    components results in only H having an odd valence.
  }
  Sum:=0;
  FOR I:=C TO P DO IF ODD(Valence[I]) THEN Sum:=Sum+Moles[I];
  OK:=OK AND (NOT ODD(Sum));
  {
  ------------------------------------
  Senior Rule #2 for CHONSPME formulae
  ------------------------------------
    The sum of the valences must equal or exceed two times the maximum
    valence.  Components don't work here.  For example, H-C≡N: passes this
    test when elemental valences of 4, 1, and 3 are used for C, H, and N.  It
    fails the test when component valences of 4 and 2 are used for C and NH.
    NaC≡N: passes the test when elemental valences of 4, 1, and 3 are used
    for C, Na, and N.  It fails the test when component valences of 4, 2, and
    0 are used for C, NH, and Na_H.

    The problem is overcome if the hidden two valences inside multiatomic
    components NH and PH are counted explicitly.
  }
  Sum:=0;
  FOR I:=C TO P DO
  CASE I OF
    N, P: Sum:=Sum+Moles[I]*Valence[I]+Moles[I]*2;
    ELSE  Sum:=Sum+Moles[I]*Valence[I];
  END;
  OK:=OK AND (Sum >= 2*Valence[C]);
  {
  ------------------------------------
  Senior Rule #3 for CHONSPME formulae
  ------------------------------------
```

```
    The difference between the maximum and minimum number of bonds that
    can exist in a molecular formula is known as unsaturation (U) or double
    bond equivalents (DBE), and U (or DBE) must be >= 0.

    Bmax = SUM(Moles[i]*Valence[i])/2

    Bmin = SUM(Moles[i]) - 1

    U = Bmax - Bmin = [SUM(Moles[i]*Valence[i]) - SUM(Moles[i]*2) + 2]/2

    U = [SUM(Moles[i]*(Valence[i] - 2)) + 2]/2

    2U = [SUM(Moles[i]*(Valence[i] - 2)) + 2] >= 0
  }
  Sum:=2;
  FOR I:=C TO P DO Sum:=Sum+Moles[I]*(Valence[I]-2);
  OK:=OK AND (Sum >=0);
  {
  -------------------------------
  Other compositional constraints
  -------------------------------

  Moles[O] should be allowed to be as large as Moles[C]+2 in CHO molecules.
  Organic molecules containing N, S, and P could possibly be organic
nitrates,
  organic sulfates, and organic phosphates, so the upper limit for Moles[O]
  should be (Moles[C]+2 + 3*Moles[N]+4*Moles[P]+4*Moles[S]).
  }
  OK:=OK AND (Moles[O]<=2+Moles[C]+3*Moles[N]+4*Moles[P]+4*Moles[S]);
  Valid:=OK;
END;

{=====================================================================
The GetCoreFormula procedure is the heart of CHOFIT3_MIN.PAS.  Here the
algorithm based on low-mass moieties CH4O(-1) and C4O(-3) is applied
to the exact mass that remains after accounting for the contributions
of all non-CHO elements/isotopes.  This procedure generates all
possible combinations of C, H, and O for a given MW and subject to the
additional constraints that are contained in the Valid function.
=====================================================================}

PROCEDURE GetCoreFormula (XEM, CoreXEM: REAL; CoreRNM: WORD;
                          VAR Loop: IFormula; VAR Found: BOOLEAN);

VAR
  CoreCEM,
  XEMerror  : REAL;
  Moieties  : SHORTINT;
  Step      : Component;
  Test      : REAL;
  Attempts  : WORD;
  GoodFit   : BOOLEAN;

CONST
  MoietyMass = 0.0363855087200022; {The exact mass of the CH4O-1 moiety}
  MaxAttempts: BYTE = 20;          {The maximum number of mixing lines }
```

```
BEGIN
  {
  Look for a CHO core formula, which must have an even NM.  Don't even
  try if CoreRNM is odd, because there cannot be a CHO core formula that
  gives an odd NM.
  }
  Found:=FALSE;
  IF NOT ODD(CoreRNM) THEN
  BEGIN
    {
    Find the hydrocarbon having this NM and the maximum number of moles
    of C.
    }
    Loop[C]:=CoreRNM DIV NM[C];
    Loop[H]:=CoreRNM -Loop[C]*NM[C];
    Loop[O]:=0;
    CoreCEM:=0;
    {
    Calculate the EM of the CHO core formula.
    }
    FOR Step:=C TO O DO
      CoreCEM:=CoreCEM+Loop[Step]*EM[Step];
    {
    Now calculate the error in matching CoreCEM and CoreXEM.  Because the
    mass of CoreXEM ó Peak.XEM, this error will be somewhat larger than the
    error based on Peak.XEM.  Note that XEMerror may be positive or negative.
    }
    CASE FitMode OF
      ppm: XEMerror:=ROUNDTO(1E6*(CoreCEM-CoreXEM)/XEM,Digits);
      mDa: XEMerror:=ROUNDTO(1E3*(CoreCEM-CoreXEM),Digits);
    END;
    Attempts:=0;
    MaxAttempts:=ROUND(0.8+CoreRNM/60);
    {
    Look for a solution on the current line connecting the CH(4)O(-1) moiety
    and the current CHO core formula.  If there is no solution, move along
    line connecting the C(4)O(-3) moiety and the current CHO core formula to
    the next line connecting the CH(4)O(-1) moiety and the new guess for
    the CHO core formula.  XEMError is compared with MaxErr because 13C has
    already been removed in FindMolecularFormulae.
    }
    WHILE (ABS(XEMerror)>MaxErr) AND (Attempts<MaxAttempts) DO
    BEGIN
      {
      Calculate the number of CH(4)O(-1) moieties that are needed to give the
      current molecular formula (the max. C hydrocarbon) the same mass as the
      CHO core formula.  Test is a real number, rounded to Digits decimal
      places.
      }
      Test:=ROUNDTO((CoreXEM-CoreCEM)/MoietyMass,Digits);
      CASE FitMode OF
        ppm: GoodFit:=(ROUNDTO(1E6*ABS((CoreCEM+ROUND(Test)*MoietyMass
                      -CoreXEM)/XEM),Digits) <= MaxErr);
        mDa: GoodFit:=(ROUNDTO(1E3*ABS(CoreCEM+ROUND(Test)*MoietyMass
                      -CoreXEM),Digits) <= MaxErr);
      END;
      IF GoodFit THEN
      BEGIN
        {
        Advance along the line connecting the CH(4)O(-1) moiety and the
        current CHO core formula to the final CHO core formula.  Because
        Test is a real number, it is possible to move toward or away from
        the CH(4)O(-1) moiety.  We need to be sure that no negative numbers
        will be generated by adding or subtracting CH(4)O(-1).
        }
        Moieties:=ROUND(Test);
        IF (Loop[C]+Moieties >= Low[C]) AND (Loop[H]+4*Moieties >= Low[H])
          AND (Loop[O]-Moieties >= Low[O]) THEN
        BEGIN
          Loop[C]:=Loop[C]+Moieties;
          Loop[H]:=Loop[H]+4*Moieties;
          Loop[O]:=Loop[O]-Moieties;
        END
        ELSE Attempts:=MaxAttempts;
      END
      ELSE
      BEGIN
        {
        Move along line connecting the C(4)O(-3) moiety and the current CHO
        core formula to the next line connecting the CH(4)O(-1) moiety and
        the new guess for the CHO core formula.  For molecular formulae
        with EM ó 1000 Da, there are no more than 20 lines to explore, so
        the WHILE loop has been limited to 20 attempts.

        Here we must also be careful not to subtract too much C, and we use
        this C(4)O(-3) moiety in single steps.
        }
        IF Loop[C]>4 THEN
        BEGIN
          Loop[C]:=Loop[C]-4;
          Loop[O]:=Loop[O]+3;
        END
        ELSE Attempts:=MaxAttempts;
      END;
      CoreCEM:=0;
      FOR Step:=C TO O DO
        CoreCEM:=CoreCEM+Loop[Step]*EM[Step];
      Test:=ROUNDTO((CoreXEM-CoreCEM)/MoietyMass,Digits);
      CASE FitMode OF
        ppm: XEMerror:=ROUNDTO(1E6*(CoreCEM-CoreXEM)/XEM,Digits);
        mDa: XEMerror:=ROUNDTO(1E3*(CoreCEM-CoreXEM),Digits);
      END;
      Attempts:=Attempts+1;
    END;
    Found:=Valid(Loop) AND (ABS(XEMerror)<=MaxErr);
  END;
END;

{=====================================================================
FindMolecularFormulae assigns all possible molecular formulae to each
mass in the input file, subject to user constraints on the moles of
C, H, O, N, S, P, M, E, and Z.  This particular version uses
conventional loops for N, S, P, M, E, and Z.  C, H, and O are found
simultaneously in GetCoreFormula by use of the new algorithm we have
developed based on low-mass moieties (LMM's).
=====================================================================}

PROCEDURE FindMolecularFormulae;

VAR
  Loop      : IFormula;
  Step      : Component;
  CoreXEM   : REAL;
  CoreRNM   : WORD;
  Formula_OK: BOOLEAN;

BEGIN
  {
  Initialize some variables that apply to the whole mass list.
  }
  BaseRec:=NIL;
  LastRec:=NIL;
  TotalFormulae:=0;
  Peak.ID:=0;
  {
  Start processing the input file.
  }
  ASSIGN (DevI,InPath+InFile);
  RESET (DevI);
  WHILE NOT EOF(DevI) DO
  BEGIN
    WITH Peak DO
    BEGIN
      {
      Initialize some variables that apply to this peak.
      }
      Fit:=FALSE;
      Formula_OK:=FALSE;
      CoreXEM:=0;
      CoreRNM:=0;
      FOR Step:=C TO Z DO
      BEGIN
        Moles[Step]:=0;
        Loop[Step]:=0;
      END;
      PeakRec:=NIL;
      ID:=SUCC(ID);
      {
      Now read some data...
      }
      READLN (DevI, IEM);
      IF ID>1 THEN
      BEGIN
        WRITE (#8#8#8#8#8#8);
        WRITE (ID:6);
      END
      ELSE WRITE ('Processing ID ',ID:6);
      {
      Convert the (presumably) singly-charged ion to a molecule.
      }
      IF IonMode=Negative THEN XEM:=IEM+Proton ELSE XEM:=IEM-Proton;
      {
      XEM is now the mass of the neutral molecule.

      Do the calculations only for LowMW <= XEM <= HighMW, rounding to the
      nearest integer masses.  Because LowMW and HighMW are integers and
      exact mass can round to nominal mass plus one, the upper limit is
      HighMW+1.
      }
      IF (ROUND(XEM) >= LowMW) AND (ROUND(XEM) <= (HighMW+1)) THEN
      BEGIN
        {
        Start looping through the Components.
        }
        Loop[C]:=Low[C];
        Loop[H]:=Low[H];
        Loop[O]:=Low[O];
        Loop[Z]:=Low[Z];
        REPEAT
          IF Loop[Z]>1 THEN XEM:=XEM*Loop[Z]/(Loop[Z]-1);
          Loop[M]:=Low[M];
          REPEAT
            Loop[P]:=Low[P];
            REPEAT
              Loop[S]:=Low[S];
              REPEAT
                Loop[N]:=Low[N];
                REPEAT
                  Loop[E]:=Low[E];
                  REPEAT
                    {
                    Strip off the exact mass of all loop constituents to
                    yield the CHO core of this molecular formula.
                    }
                    CoreXEM:=Peak.XEM;
                    FOR Step:=N TO E DO CoreXEM:=CoreXEM-Loop[Step]*EM[Step];
                    {
                    CoreXEM is required to be as large as the EM of CH(4).
                    }
                    IF CoreXEM>=EM[C]+4*EM[H] THEN
                    BEGIN
                      CoreRNM:=ROUND(CoreXEM);
                      Formula_OK:=FALSE;
                      {
                      For EM ó 1000 Da, only H can cause a "rounding up"
                      error, which occurs first at C(31)H(64), for which
                      the EM is 436.500802 Da.  If EM >= 436.500802 Da, an
                      even CoreRNM could actually be odd and vice versa.
                      }
                      IF NOT ODD(CoreRNM) THEN
                        GetCoreFormula (XEM, CoreXEM, CoreRNM, Loop,
Formula_OK)
                      ELSE
                      IF ODD(CoreRNM) AND (CoreXEM>=31*EM[C]+64*EM[H]) THEN
                        GetCoreFormula (XEM, CoreXEM, CoreRNM-1, Loop,
Formula_OK);
                      {
                      If this is a good formula, the residual mass should
                      be zero (or nearly so).  The error of the fit is
                      calculated either in mDa or ppm.
                      }
                      IF Formula_OK THEN
                      BEGIN
                        {
                        Re-calculate the CEM and CNM using updated Loop[]
                        values.
                        }
                        CEM:=0;
                        CNM:=0;
                        FOR Step:=C TO E DO
                        BEGIN
                          CEM:=CEM+Loop[Step]*EM[Step];
                          CNM:=CNM+Loop[Step]*NM[Step];
                        END;
                        CASE FitMode OF
                          ppm: XEMerr:=ROUNDTO(1E6*(CEM-XEM)/XEM,Digits);
                          mDa: XEMerr:=ROUNDTO(1E3*(CEM-XEM),Digits);
                        END;
                        Formula_OK:=(ABS(XEMerr) <= MaxErr);
                      END
                      ELSE Formula_OK:=FALSE;
                    END
                    ELSE Formula_OK:=FALSE;
                    {
                    All of the following calculations are only done if the
                    formula is valid.
                    }
                    IF Formula_OK THEN
                    BEGIN
                      {
                      CNM:=0;
                      FOR Step:=C TO E DO CNM:=CNM+Loop[Step]*NM[Step];
                      }
                      {
                      Transfer the molecular formula from Loop to Moles.
                      }
                      FOR Step:=C TO Z DO Moles[Step]:=Loop[Step];
                      Fit:=TRUE;
                      {
                      Store the valid solution in a linked list.
                      }
                      BuildFitRec (Peak,LastRec,NIL,LastRec);
                    END;
                    {
                    The E loop is terminated when E>1.
                    }
                    Loop[E]:=SUCC(Loop[E]);
                  UNTIL (Loop[E]>High[E]);
                  {
```

```
                  The N loop is terminated when N>HighN.
                  }
                  Loop[N]:=SUCC(Loop[N]);
                UNTIL (Loop[N]>High[N]);
                {
                The S loop is terminated when S>HighS.
                }
                Loop[S]:=SUCC(Loop[S]);
              UNTIL (Loop[S]>High[S]);
              {
              The P loop is terminated when P>HighP.
              }
              Loop[P]:=SUCC(Loop[P]);
            UNTIL (Loop[P]>High[P]);
            {
            The M loop is terminated when M>HighM.
            }
            Loop[M]:=SUCC(Loop[M]);
          UNTIL (Loop[M]>High[M]);
          {
          The Z loop is used when no solution can be found for Z=1.  The Z
          loop is terminated when Z>HighZ.
          }
          IF Fit THEN Loop[Z]:=High[Z];
          Loop[Z]:=SUCC(Loop[Z]);
        UNTIL (Loop[Z]>High[Z]);
        {
        Insert an "empty" record if no molecular formula could be fit to the
        peak.
        }
      END;
      IF (NOT Fit) THEN
      BEGIN
        {
        Reset XEM to the mass of a molecule that was detected as a singly
        charged ion.
        }
        XEM:=XEM/High[Z];
        CEM:=0;
        CNM:=0;
        XEMerr:=0;
        BuildFitRec (Peak,LastRec,NIL,LastRec);
      END;
    END;
  END;
  CLOSE (DevI);
END;


{======================================================================
WriteOutputFile writes the output of CHOFIT3_MIN.PAS to a text file.
======================================================================}

PROCEDURE WriteOutputFile (VAR BaseRec: FitPtr);

VAR
```

```
  ThisRec: FitPtr;
  I      : Component;
  HCount : BYTE;
  Q      : SHORTINT;
  Count  : LONGWORD;
  Txt    : STRING[40];

BEGIN
  WRITELN;
  WRITE ('Writing the output file...');
  ASSIGN (DevO, OutPath+OutFile);
  REWRITE (DevO);
  ThisRec:=BaseRec;
  {
  Write the column headings to the output file.
  }
  WRITELN (DevO,'Program Name: ':15,Version);
  WRITELN (DevO,'Input File : ':15,InPath+InFile);
  WRITELN (DevO,'Output File : ':15,OutPath+OutFile);
  WRITELN (DevO);
  WRITE   (DevO,'ID':6,'IEM':12,'XEM':12,'CEM':12,'CNM':6);
  WRITE   (DevO,'13C':6,'12C':6,'1H':6);
  WRITE   (DevO,'16O':6,'14N':6,'32S':6,'31P':6,'23Na':6,'Z':6);
  WRITE   (DevO,'XEMerr':10);
  WRITELN (DevO);
  WHILE ThisRec<>NIL DO
  BEGIN
    WITH ThisRec^.Peak DO
    BEGIN
        WRITE (DevO,ID:6,IEM:12:6,XEM:12:6,CEM:12:6,CNM:6);
      WRITE (DevO,Moles[E]:6,(Moles[C]-Moles[E]):6);
      WRITE (DevO,(Moles[H]+Moles[N]+Moles[P]-Moles[M]):6);
      FOR I:=O TO M DO WRITE (DevO, Moles[I]:6);
      IF IonMode=Negative THEN
        WRITE (DevO,-Moles[Z]:6) ELSE WRITELN (DevO,Moles[Z]:6);
      WRITE (DevO,XEMerr:10:6);
      WRITELN (DevO);
    END;
    ThisRec:=ThisRec^.Next;
  END;
  CLOSE (DevO);
  WRITELN;
  WRITE ('Press ENTER to close the program...');
  READLN;
END;

{=====================================================================
Main program in CHOFIT3_MIN.PAS
=====================================================================}

BEGIN
  IF ParamCount = 8 THEN
  BEGIN
    GetDir (0,InPath);
    InPath:=InPath+'\';
    InFile:=Paramstr(1);
    IF POS('.',InFile)=0 THEN InFile:=InFile+'.dat';
IF Exists (InPath+InFile) THEN
      BEGIN
        OutPath:=InPath;
        OutFile:=Paramstr(2);
        IF POS('.',OutFile)=0 THEN OutFile:=OutFile+'.fit';
        Val(Paramstr(3),LowMW);
        IF (LowMW < MinMW) OR (LowMW > MaxMW) THEN LowMW:=MinMW;
        Val(Paramstr(4),HighMW);
        IF (HighMW < LowMW) OR (HighMW > MaxMW) THEN HighMW:=MaxMW;
        Val(Paramstr(5),High[N]);
        IF (High[N] < Min[N]) OR (High[N] > Max[N]) THEN High[N]:=Max[N];
        Val(Paramstr(6),High[S]);
        IF (High[S] < Min[S]) OR (High[S] > Max[S]) THEN High[S]:=Max[S];
        Val(Paramstr(7),High[P]);
        IF (High[P] < Min[P]) OR (High[P] > Max[P]) THEN High[P]:=Max[P];
        Val(Paramstr(8),High[E]);
        IF (High[E] < Min[E]) OR (High[E] > Max[E]) THEN High[E]:=Max[E];
        WRITELN;
        WRITELN;
        WRITELN ('Program: ',Version);
        WRITELN ('InFile : ',InPath+InFile);
        WRITELN ('OutFile: ',OutPath+OutFile);
        WRITELN ('Low MW : ',LowMW:6, ' ':2,'High MW : ',HighMW:6);
        WRITELN ('Low C  : ',Low[C]:6,' ':2,'High C  : ',High[C]:6);
        WRITELN ('Low H  : ',Low[H]:6,' ':2,'High H  : ',High[H]:6);
        WRITELN ('Low O  : ',Low[O]:6,' ':2,'High O  : ',High[O]:6);
        WRITELN ('Low N  : ',Low[N]:6,' ':2,'High N  : ',High[N]:6);
        WRITELN ('Low S  : ',Low[S]:6,' ':2,'High S  : ',High[S]:6);
        WRITELN ('Low P  : ',Low[P]:6,' ':2,'High P  : ',High[P]:6);
        WRITELN ('Low Na : ',Low[M]:6,' ':2,'High Na : ',High[M]:6);
        WRITELN ('Low 13C: ',Low[E]:6,' ':2,'High 13C: ',High[E]:6);
        FindMolecularFormulae;
        WriteOutputFile (BaseRec);
        DeleteFitRecs (BaseRec);
        WRITELN;
        WRITELN;
      END
      ELSE
    BEGIN
      WRITE ('Input file does not exist.  Press ENTER to exit and try
again.');
      READLN;
      EXIT;
    END;
  END
  ELSE
  BEGIN
    WRITELN;
    WRITELN;
    WRITELN ('Please launch this program from a command prompt using the
syntax:');
    WRITELN;
    WRITELN ('ProgramName.exe InputFileName OutputFileName Low_MW High_MW N S
P 13C');
    WRITELN;
    WRITELN ('where InputFileName and OutputFileName may be entered
with/without ');
    WRITELN ('extensions.  If a file extension is omitted, then a default
extension');
    WRITELN ('of ".dat" will be appended to InputFileName and a default
extension of');
    WRITELN ('".fit" will be appended to OutputFileName.');
    WRITELN;
    WRITELN ('Low_MW and High_MW are the limits of nominal mass that will be
evaluated.');
    WRITELN;
    WRITELN ('N, S, P, and 13C are the upper limits for the number of these
atoms that');
    WRITELN ('can be used in a molecular formula.  The maximum values are
N=10, S=6,');
    WRITELN ('P=4, and 13C=1. Lower limits for these elements/isotopes are
all zero.');
    WRITELN;
    WRITELN;
    WRITE ('Press ENTER to exit and try again...');
    READLN;
    WRITELN;
    WRITELN;
  END;
END.
```