CHEM 225 Name: _____

Programming Exercise 2

Complete the following problems.
This assignment is due by 11:59 pm on March 15, 2019.

This homework assignment will introduce you to packed storage of matrices. In numerical codes, particularly those that rely on BLAS and LAPack libraries, it is common to work with matrices in a so-called *packed* storage form. A starting point to understand the use of packed storage form is to understand row- and column-based packed storage form of general matrices. Consider the following matrix, $\mathbf{A}$,

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

In a *row* linear packed form, the matrix $\mathbf{A}$ is represented as the vector $\mathbf{a}_{row}$; while in *column* linear packed form, the matrix is represented as the vector $\mathbf{a}_{col}$. *Viz*.

$$\mathbf{a}_{row} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix} \quad , \quad \mathbf{a}_{col} = \begin{pmatrix} 1 \\ 4 \\ 7 \\ 2 \\ 5 \\ 8 \\ 3 \\ 6 \\ 9 \end{pmatrix}$$

Perhaps the most common form of packed storage matrices in quantum chemistry codes involves symmetric matrices. An $N \times N$ packed symmetric matrix includes only the $N(N+1)/2$ unique matrix elements. When using a matrix stored in this form, it is critical that the packed storage form is fully understood and used consistently throughout a program. Specifically, is must be known if the packed storage form works on the top or bottom half of the matrix and whether the storage given in column- or row-wise form. For example, consider the symmetric matrix, $\mathbf{B}$, where the lower-triangle portion of the matrix is given in boldface.

$$\mathbf{B} = \begin{pmatrix} \mathbf{1} & 2 & 3 \\ \mathbf{2} & \mathbf{4} & 5 \\ \mathbf{3} & \mathbf{5} & \mathbf{6} \end{pmatrix}$$

The column ($\mathbf{b}_{col-lower}$) and row ($\mathbf{b}_{row-lower}$) based lower-triangle forms are

$$\mathbf{b}_{col-lower} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \quad , \quad \mathbf{b}_{row-lower} = \begin{pmatrix} 1 \\ 2 \\ 4 \\ 3 \\ 5 \\ 6 \end{pmatrix}$$

The column ($\mathbf{b}_{col-lower}$) and row ($\mathbf{b}_{row-lower}$) based upper-triangle forms are

$$\mathbf{b}_{col-upper} = \begin{pmatrix} 1 \\ 2 \\ 4 \\ 3 \\ 5 \\ 6 \end{pmatrix} \quad , \quad \mathbf{b}_{row-upper} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$$

Note that symmetric matrix column-lower and row-upper packed storage forms are equivalent. Likewise, symmetric matrix row-lower and column-upper pack storage forms are equivalent.

1. Write and compile a program named `pgrm_02_01.exe` that takes the name of a file containing a list of numbers that make up a square matrix given in linear packed form, converts the packed form array into a full rank-2 array and then prints the final matrix. Program specifics include:

   - The program should expect and accept only 1 command line argument, the name of the input file. For example, if the name of the user-provided input file is `matrix.inp` the command line argument will be '`Homework-05-01.exe matrix.inp`'.
   - The program will include a subroutine called `Packed2Matrix_ColumnWise`, which should include an argument list of `M,N,ArrayIn,AMatOut` where `M` and `N` are input arguments giving the sizes of the two dimensions of the full matrix, `ArrayIn` is an input argument giving the rank-1 array giving the pack storage form of the matrix, and `AMatOut` is an output argument returning the full (un-package) matrix. The conversion of `ArrayIn` to `ArrayOut` is carried out taking the linear packed storage form to be in *row* format.
   - The program will include a subroutine called `Packed2Matrix_RowWise`, which should include an argument list of `(M,N,ArrayIn,AMatOut)` where `M` and `N` are input arguments giving the sizes of the two dimensions of the full matrix, `ArrayIn` is an input argument giving the rank-1 array giving the pack storage form of the matrix, and `AMatOut` is an output argument returning the full (un-package) matrix. The conversion of `ArrayIn` to `ArrayOut` is carried out taking the linear packed storage form to be in *row* format.
   - After reading the user-provided file giving the matrix size and linear packed storage form, the program will call subroutine `Packed2Matrix_ColumnWise` and then call a routine to print the full-matrix showing that the column-linear pack storage form was correctly converted. The program will then call subroutine `Packed2Matrix_RowWise` and then call a routine to print the full-matrix showing that the column-linear pack storage form was correctly converted.

   An example input file named `input_02_01.txt` is provided in your GitHub repository. In the same directory, there is also a general full matrix printing subroutine provided in the file `print_matrix_full_real.f90`. The source code in that file should be copied and included within your source code file.

   Following the code design requirements above, the required main program should be (or be quite similar to) the following (also included in your GitHub repository):

```
      Program pgrm_02_01
!
!     This program reads a file name from the command line, opens that file, and
!     loads a packed/linearized form of a square matrix. Then, the packed matrix
!     is expanded assuming a row-packed form and printed. Finally, the packed
!     matrix is expanded as a column-packed form and printed.
!
!     The input file is expected to have the leading dimension (an integer
!     NDim) of the matrix on the first line. The next NDim*NDim lines each
!     have one real number each given.
!
      Implicit None
      Integer,Parameter::IIn=10
      Integer::IError,NDim,i,j
      Real,Dimension(:),Allocatable::Array_Input
      Real,Dimension(:,:),Allocatable::Matrix
      Character(Len=256)::FileName
!
!     Begin by reading the input file name from the command line. Then,
!     open the file and read the input array, Array_Input.
!
      Call Get_Command_Argument(1,FileName)
      Open(Unit=IIn,File=TRIM(FileName),Status='OLD',IOStat=IError)
      If(IError.ne.0) then
        Write(*,*)' Error opening input file.'
        STOP
      endIf
      Read(IIn,*) NDim
      Allocate(Array_Input(NDim*NDim),Matrix(NDim,NDim))
!
! ************************************************************************
! WRITE CODE HERE TO READ THE ARRAY ELEMENTS FROM THE INPUT FILE.
! ************************************************************************
!
!
!     Convert Array_Input to Matrix and print the matrix.
!
      Write(*,*)' The matrix expanded according to a row-wise ', &
        'linear packed format:'
      Call Packed2Matrix_RowWise(NDim,NDim,Array_Input,Matrix)
      Call Print_Matrix_Full_Real(Matrix,NDim,NDim)
      Write(*,*)' The matrix expanded according to a column-wise ', &
        'linear packed format:'
      Call Packed2Matrix_ColumnWise(NDim,NDim,Array_Input,Matrix)
      Call Print_Matrix_Full_Real(Matrix,NDim,NDim)
!
      End Program pgrm_02_01
```

```
      Subroutine Packed2Matrix_ColumnWise(M,N,ArrayIn,AMatOut)
!
!     This subroutine accepts an array, ArrayIn, that is M*N long. It then
!     takes those elements and converts them to the M-by-N matrix AMatOut
!     such that the elements in ArrayIn are interpreted as a packed
!     column-wise form of the matrix AMatOut.
!
      Implicit None
      Integer,Intent(In)::M,N
      Real,Dimension(N*M),Intent(In)::ArrayIn
      Real,Dimension(M,N),Intent(Out)::AMatOut
!
      Integer::i,j,k
!
!     Loop through the elements of AMatOut and fill them appropriately from
!     Array_Input.
!
!
! ************************************************************************
! WRITE CODE HERE TO READ THE ARRAY ELEMENTS FROM THE INPUT FILE.
! ************************************************************************
!
!
      Return
      End Subroutine Packed2Matrix_ColumnWise


      Subroutine Packed2Matrix_RowWise(M,N,ArrayIn,AMatOut)
!
!     This subroutine accepts an array, ArrayIn, that is M*N long. It then
!     takes those elements and converts them to the M-by-N matrix AMatOut
!     such that the elements in ArrayIn are interpreted as a packed
!     row-wise form of the matrix AMatOut.
!
      Implicit None
      Integer,Intent(In)::M,N
      Real,Dimension(N*M),Intent(In)::ArrayIn
      Real,Dimension(M,N),Intent(Out)::AMatOut
!
      Integer::i,j,k
!
!     Loop through the elements of AMatOut and fill them appropriately from
!     Array_Input.
!
!
! ************************************************************************
! WRITE CODE HERE TO READ THE ARRAY ELEMENTS FROM THE INPUT FILE.
```

```
! ************************************************************************
!
!
      Return
      End Subroutine Packed2Matrix_RowWise
```

As an example, the command

```
./pgrm_02_01.exe input_02_01.txt
```

should give the output:

```
 The matrix expanded according to a row-wise linear packed format:
                1             2             3
     1      1.000000      2.000000      3.000000
     2      4.000000      5.000000      6.000000
     3      7.000000      8.000000      9.000000
 The matrix expanded according to a column-wise linear packed format:
                1             2             3
     1      1.000000      4.000000      7.000000
     2      2.000000      5.000000      8.000000
     3      3.000000      6.000000      9.000000
```

2. Write and compile a program named `pgrm_02_01.exe` that takes the name of a file containing a list of numbers that comprise a packed storage symmetric matrix, converts and prints the matrix assuming column-wise lower-triangle symmetric packed storage, and then converts and prints the matrix assuming column-wise upper-triangle symmetric packed storage.

   An example input file named `input_02_02.txt` is provided on in your GitHub repository. In the same directory, there is also a general full matrix printing subroutine provided in the file `print_matrix_full_real.f90`. The source code in that file should be copied and included within your source code file.

   Your program should be (or be quite similar to) the following:

```
      Program pgrm_02_02
!
!     This program reads a file name from the command line, opens that
!     file, and loads a packed form of a symmetric matrix. Then, the packed
!     matrix is expanded assuming a column-wise lower-triangle packed form
!     and printed. Finally, the packed matrix is expanded as a column-wise
!     upper-triangle packed form and printed.
!
!     The input file is expected to have the leading dimension (an integer
!     NDim) of the matrix on the first line. The next (NDim*(NDim+1))/2
!     lines each have one real number each given.
!
```

```
!
      Implicit None
      Integer,Parameter::IIn=10
      Integer::IError,NDim,i,j
      Real,Dimension(:),Allocatable::Array_Input
      Real,Dimension(:,:),Allocatable::Matrix
      Character(Len=256)::FileName
!
!     Begin by reading the input file name from the command line. Then,
!     open the file and read the input array, Array_Input.
!
      Call Get_Command_Argument(1,FileName)
      Open(Unit=IIn,File=TRIM(FileName),Status='OLD',IOStat=IError)
      If(IError.ne.0) then
        Write(*,*)' Error opening input file.'
        STOP
      endIf
      Read(IIn,*) NDim
      Allocate(Array_Input((NDim*(NDim+1))/2),Matrix(NDim,NDim))
!
! ************************************************************************
! WRITE CODE HERE TO READ THE ARRAY ELEMENTS FROM THE INPUT FILE.
! ************************************************************************
!
!
!     Convert Array_Input to Matrix and print the matrix.
!
      Write(*,*)' The matrix loaded (column-wise) lower-tri packed:'
      Call SymmetricPacked2Matrix_LowerPac(NDim,Array_Input,Matrix)
      Call Print_Matrix_Full_Real(Matrix,NDim,NDim)
      Write(*,*)' The matrix loaded (column-wise) upper-tri packed:'
      Call SymmetricPacked2Matrix_UpperPac(NDim,Array_Input,Matrix)
      Call Print_Matrix_Full_Real(Matrix,NDim,NDim)
!
      End Program pgrm_02_02


      Subroutine SymmetricPacked2Matrix_LowerPac(N,ArrayIn,AMatOut)
!
!     This subroutine accepts an array, ArrayIn, that is (N*(N+1))/2 long.
!     It then converts that form to the N-by-N matrix AMatOut taking
!     ArrayIn to be in lower-packed storage form. Note: The storage mode
!     also assumes the lower-packed storage is packed by columns.
!
      Implicit None
      Integer,Intent(In)::N
      Real,Dimension((N*(N+1))/2),Intent(In)::ArrayIn
      Real,Dimension(N,N),Intent(Out)::AMatOut
```

```
!
      Integer::i,j,k
!
!     Loop through the elements of AMatOut and fill them appropriately from
!     Array_Input.
!
!
! ************************************************************************
! WRITE CODE HERE TO READ THE ARRAY ELEMENTS FROM THE INPUT FILE.
! ************************************************************************
!
!
      Return
      End Subroutine SymmetricPacked2Matrix_LowerPac


      Subroutine SymmetricPacked2Matrix_UpperPac(N,ArrayIn,AMatOut)
!
!     This subroutine accepts an array, ArrayIn, that is (N*(N+1))/2 long.
!     It then converts that form to the N-by-N matrix AMatOut taking
!     ArrayIn to be in upper-packed storage form. Note: The storage mode
!     also assumes the upper-packed storage is packed by columns.
!
      Implicit None
      Integer,Intent(In)::N
      Real,Dimension((N*(N+1))/2),Intent(In)::ArrayIn
      Real,Dimension(N,N),Intent(Out)::AMatOut
!
      Integer::i,j,k
!
!     Loop through the elements of AMatOut and fill them appropriately from
!     Array_Input.
!
!
! ************************************************************************
! WRITE CODE HERE TO READ THE ARRAY ELEMENTS FROM THE INPUT FILE.
! ************************************************************************
!
!
      Return
      End Subroutine SymmetricPacked2Matrix_UpperPac
```

As an example, the command `./pgrm_02_02.exe input_02_02.txt` should give the output:

```
 The matrix loaded (column-wise) lower-tri packed:
                1             2             3             4
```

```
      1      1.000000      2.000000      3.000000      4.000000
      2      2.000000      5.000000      6.000000      7.000000
      3      3.000000      6.000000      8.000000      9.000000
      4      4.000000      7.000000      9.000000     10.000000
 The matrix loaded (column-wise) upper-tri packed:
                 1             2             3             4
      1      1.000000      2.000000      4.000000      7.000000
      2      2.000000      3.000000      5.000000      8.000000
      3      4.000000      5.000000      6.000000      9.000000
      4      7.000000      8.000000      9.000000     10.000000
```

3. This problem provides an introduction to using numeric libraries, specifically the BLAS and LAPack libraries. As a pre-cursor exercise, please read the descriptions of these packages online at the addresses:

   https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms

   http://www.netlib.org/blas/

   https://en.wikipedia.org/wiki/LAPACK

   http://www.netlib.org/lapack/

   The PGI compiler (and most other compilers) ship an optimized version of the BLAS and LAPack libraries. To link a program to these included libraries, simply include the switches `-lblas -llapack` with the compilation of the program. For example, to compile a program named `MyProgram.exe` from a source code file named `MyProgram.f90` that calls BLAS and LAPack routines, one can use the compile command:

   ```
   pgfortran -lblas -llapack -o pgrm_02_03.exe pgrm_02_03.f90.
   ```

   Write and compile a program named `pgrm_02_03.exe` that takes the name of a file containing a list of numbers that comprise a packed storage symmetric matrix, converts and prints the matrix assuming column-wise lower-triangle symmetric packed storage, uses the LAPack routine `SSPEV` and then converts and prints the matrix assuming column-wise upper-triangle symmetric packed storage.

   Your program should be (or be quite similar to) the following:

```
      Program pgrm_02_03
!
!     This program reads a file name from the command line, opens that
!     file, and loads a packed form of a symmetric matrix. Then, the packed
!     matrix is expanded assuming a column-wise lower-triangle form and
!     printed. Finally, the matrix is diagonalized using the LAPack routine
!     SSPEV. The eigenvalues and eigenvectors are printed.
!
!     The input file is expected to have the leading dimension (an integer
!     NDim) of the matrix on the first line. The next (NDim*(NDim+1))/2
!     lines each have one real number each given.
!
```

```
!
      Implicit None
      Integer::IIn,IError,NDim,i,j
      Real,Dimension(:),Allocatable::Array_Input,EVals,Temp_Vector
      Real,Dimension(:,:),Allocatable::Matrix,EVecs,Temp_Matrix
      Character(Len=256)::FileName
!
!     Begin by reading the input file name from the command line. Then,
!     open the file and read the input array, Array_Input.
!
      Call Get_Command_Argument(1,FileName)
      Open(Unit=IIn,File=TRIM(FileName),Status='OLD',IOStat=IError)
      If(IError.ne.0) then
        Write(*,*)' Error opening input file.'
        STOP
      endIf
      Read(IIn,*) NDim
      Allocate(Array_Input((NDim*(NDim+1))/2),Matrix(NDim,NDim))
      Allocate(EVals(NDim),EVecs(NDim,NDim),Temp_Vector(3*NDim))
      Allocate(Temp_Matrix(NDim,NDim))
!
! ************************************************************************
! WRITE CODE HERE TO READ THE ARRAY ELEMENTS FROM THE INPUT FILE.
! ************************************************************************
!
      Close(Unit=IIn)
!
!     Convert Array_Input to Matrix and print the matrix.
!
      Write(*,*)' The matrix loaded (column) lower-triangle packed:'
      Call SymmetricPacked2Matrix_LowerPac(NDim,Array_Input,Matrix)
      Call Print_Matrix_Full_Real(Matrix,NDim,NDim)
      Call SSPEV('***','***',NDim,Array_Input,EVals,EVecs,NDim,  &
        Temp_Vector,IError)
      If(IError.ne.0) then
        Write(*,*)' Failure in DSPEV.'
        STOP
      endIf
      Write(*,*)' EVals:'
      Call Print_Matrix_Full_Real(RESHAPE(EVals,(/1,NDim/)),1,NDim)
      Write(*,*)' EVecs:'
      Call Print_Matrix_Full_Real(EVecs,NDim,NDim)
!
      End Program pgrm_02_03
```

Note that the arguments to `Subroutine SSPEV` given by `***` must be replaced with appropriate characters values. View the comments to `SSPEV` to determine appropriate values for these

arguments.

As an example, the command `./pgrm_02_03.exe input_02_03.inp` should give the output:

```
The matrix loaded (column) lower-triangle packed:
                1             2             3             4
     1      1.900000      0.800000      1.300000      2.000000
     2      0.800000      3.200000      1.000000      2.500000
     3      1.300000      1.000000      4.700000      1.100000
     4      2.000000      2.500000      1.100000      5.200000
EVals:
                1             2             3             4
     1      0.692586      1.776067      3.984381      8.546966
EVecs:
                1             2             3             4
     1      0.806119     -0.475698     -0.048653      0.348593
     2      0.294005      0.811609      0.213511      0.457456
     3     -0.205707      0.059523     -0.874673      0.434844
     4     -0.470552     -0.333856      0.432427      0.692912
```

As a second example, the command `./pgrm_02_03.exe input_02_04.inp` should give the output:

```
The matrix loaded (column) lower-triangle packed:
                1             2             3             4
     1      2.000000      1.000000      0.000000      1.500000
     2      1.000000      1.000000      0.200000      0.000000
     3      0.000000      0.200000      2.500000      0.000000
     4      1.500000      0.000000      0.000000      2.500000
EVals:
                1             2             3             4
     1      0.034288      1.515083      2.526013      3.924615
EVecs:
                1             2             3             4
     1      0.634157     -0.386584     -0.001495      0.669624
     2     -0.667892     -0.695674      0.128503      0.231181
     3      0.054174      0.141266      0.987955      0.032455
     4     -0.385785      0.588757     -0.086193      0.705058
```

Both of these sample input files are available in your GitHub repository.