PS6

1.1

Write a program Main.f90 to read fortran_demo1/M.dat as the matrix M, and fortran_demo1/N.dat as the matrix N （below are part of the main program）

```
integer :: u,i,j
real(4), dimension(5,3) :: M
real(4), dimension(3,5) :: N
real(4), dimension(5,5) :: MN

u=50

!M Matrix
open(unit=u, file='M.dat', status='old')

read(u,*)((M(i,j),j=1,3),i=1,5)

close(u)

!display
print*,'M'

do i=1,5

        write(*,'(5f8.2)') M(i,:)

enddo

!N Matrix
open(unit=u, file='N.dat', status='old')

read(u,*)((N(i,j),j=1,5),i=1,3)

close(u)

!display
print*,'N'

do i=1,3

        write(*,'(5f8.2)') N(i,:)

enddo
```

1.2

Write a subroutine Matrix_multip.f90 to do matrix multiplication

```
subroutine Matrix_multip(a,b,c)

implicit none

real(4),dimension(5,3), intent(in) :: a
real(4),dimension(3,5), intent(in) :: b
real(4),dimension(5,5), intent(out) :: c

c=matmul(a,b)

end subroutine Matrix_multip
```

1.3

(1) Call the subroutine Matrix_multip() from Main.f90 to compute M*N; write the output to a new file MN.dat, values are in formats of f9.2

(below are part of the main program)

```
!call subroutine
call Matrix_multip(M, N, MN)

!display MN
print*,'MN'
write(*,'(5f8.2)') MN

!write MN.dat
open(unit=u, file='MN.dat', status='replace')

do i=1,5

        write(u,'(f9.2,f9.2,f9.2,f9.2,f9.2)') MN(i,:)

enddo
close(u)
```

(2) Compile the main.f90

(3) Result:

```
[ese-wurx@login03 fortran_demo1]$ gfortran Matrix_multip.f90 main.f90 -o main.x
[ese-wurx@login03 fortran_demo1]$ ./main.x
 M
   19.48   15.79   19.28
   19.28   12.92   15.86
   15.86   11.29   14.04
   11.93   18.60   18.23
   19.28   12.92   15.86
 N
    7.72    4.11    1.44    4.80    5.55
    5.55    4.80    4.04    0.59    8.58
    0.59    8.58    2.26    7.72    4.11
 MN
  249.40  229.90  193.38  206.09  229.90
  321.28  277.34  239.84  294.73  277.34
  135.42  115.80  100.18  133.52  115.80
  251.66  222.61  191.18  208.97  222.61
  322.83  283.04  242.60  300.72  283.04
```

(4) The MN.dat:

```
  GNU nano 2.3.1                                File: MN.dat

   249.40    321.28    135.42    251.66    322.83
   229.90    277.34    115.80    222.61    283.04
   193.38    239.84    100.18    191.18    242.60
   206.09    294.73    133.52    208.97    300.72
   229.90    277.34    115.80    222.61    283.04
```

2.1

Write a module Declination_angle that calculates the declination angle on a given date

```
module Declination_angle

implicit none

        integer::d
        real(8)::a,b,pi

contains
        subroutine calculate1()
        pi=3.14159265358979
        write(*,*) 'input the number of days since January 1st d'
        read(*,*) d

        b=COS(pi/180*(360/365.24)*(d+10)+(360/pi)*0.0167*SIN((pi/180*360/365.24)*(d-2)))

        a=(ASIN(SIN(-23.44*pi/180)*b))*180/pi

        end subroutine calculate1


end module Declination_angle
```

2.2

Write a module Solar_hour_angle that calculates the solar hour angle in a given location for a given date and time

```
module Solar_hour_angle

        real(4)::h,LST

        contains
        subroutine calculate2()

        write(*,*) 'input the local solar time (in minutes)  LST'
        read(*,*) LST

        h=15*((LST/60)-12)

        end subroutine calculate2

end module Solar_hour_angle
```

2.3

Write a main program (Solar_elevation_angle.f90) that uses module Declination_angle and Solar_hour_angle to calculate and print the SEA in a given location for a given date and time.

```
program Solar_elevation_angle

use declination_angle
use solar_hour_angle

implicit none

real(4)::SEA,L

write(*,*)'input latitude L'
read(*,*) L

call calculate1()

call calculate2()

SEA=(ASIN(SIN(L*pi/180)*SIN(a*pi/180)+COS(L*pi/180)*COS(a*pi/180)*COS(h*pi/180)))*180/pi

print*, "declination_angle = ",a
print*, "solar_hour_angle = ",h
print*,"Solar_elevation_angle=",SEA

end program Solar_elevation_angle
```

2.4

(1) Create a library (libsea.a) that contains Declination_angle.o and Solar_hour_angle.o

```
[ese-wurx@login03 fortran_demo1]$ gfortran -c Solar_elevation_angle.f90
[ese-wurx@login03 fortran_demo1]$ gfortran -c Solar_hour_angle.f90
[ese-wurx@login03 fortran_demo1]$ gfortran -c Declination_angle.f90
```

```
[ese-wurx@login03 fortran_demo1]$ ar rcvf libsea.a Declination_angle.o Solar_hour_angle.o
a - Declination_angle.o
a - Solar_hour_angle.o
```

```
-rw-r--r-- 1 ese-wurx ese-ouycc  6554 Dec 21 23:47 libsea.a
```

(2) Compile Solar_elevation_angle.f90 using libsea.a.

```
[ese-wurx@login03 fortran_demo1]$ gfortran Solar_elevation_angle.f90 -o Solar_elevation_angle_lib.x -L. -lsea
```

(3) Print the SEA for Shenzhen (22.542883N, 114.062996E) at 10:32 (Beijing time; UTC+8) on 2021-12-31.

Input:

1. latitude of Shenzhen: 22.542883
2. d: the number of days of 2021-12-31 since 2012-1-1 is 364
3. LST:10:32 turns into local solar time in minutes 10*60+32=632

```
[ese-wurx@login03 fortran_demo1]$ gfortran Solar_elevation_angle.f90 -o Solar_elevation_angle_lib.x -L. -lsea
[ese-wurx@login03 fortran_demo1]$ ./Solar_elevation_angle_lib.x
 input latitude L
22.542883
 input the number of days since January 1st d
364
 input the local solar time (in minutes)  LST
632
 declination_angle =   -23.415861463273444
 solar_hour_angle =   -21.9999924
 Solar_elevation_angle=   39.3060265
```