Write a program that reads two vectors from the keyboard (as two ordered triplets). The program then prints the vectors in both Cartesian \((x, y, z)\) and spherical \((r, \phi, \theta)\) representation, as well as the dot product of the two vectors, and their cross product (displayed in Cartesian coordinates). The program should continue to accept input until the user hits control-D, at which point the program simply terminates. Use modular programming so that functions or subroutines are used where appropriate (i.e., the dot product should be calculated with a function while the cross product and any coordinate transformations should be calculated with subroutines).

To make this assignment more interesting, the user should be able to enter the vectors in either Cartesian or spherical coordinates. The program should start by assuming the user wants to enter values in Cartesian coordinates. However, the program should “toggle states” and assume the user wants to enter values in spherical coordinates if there was an error reading the input. Thus, you will need to use the IOSTAT clause on the read statement to see if an error occurred. The user should be instructed to enter a non-number character to toggle the type of input that is accepted. Angles should be accepted and reported in degrees.

There are many ways a program can keep track of its “state” (i.e., keep track of how it should behave). Shown on the back is a simple program that reads a real number and prints it back out. The program also remembers the state it is in and changes state if there was an error reading the input. The total number of states is specified by the variable \(\text{numsts}\) which is initialized to five in this particular case (thus the user could toggle through five different states). You can use this program as a starting point for your solution to this assignment—modify it as you see fit. Note that this assignment just requires two states: Cartesian input or spherical input. This program is available from the class Web site (see the Assignments page). Play with it and make sure you understand its behavior before piecing in the code to do the rest of the assignment.

In case you’re rusty on your coordinate transformations and vector operations

\[
\begin{align*}
x &= r \cos \phi \sin \theta \\
y &= r \sin \phi \sin \theta \\
z &= r \cos \theta \\
r &= \sqrt{x^2 + y^2 + z^2} \\
\phi &= \tan^{-1}(y/x) \\
\theta &= \tan^{-1}(\sqrt{x^2 + y^2} / z) \\
(x_1, y_1, z_1) \cdot (x_2, y_2, z_2) &= x_1 x_2 + y_1 y_2 + z_1 z_2 \\
(x_1, y_1, z_1) \times (x_2, y_2, z_2) &= (y_1 z_2 - y_2 z_1, x_2 z_1 - x_1 z_2, x_1 y_2 - x_2 y_1)
\end{align*}
\]

Note that you have to be careful with the arctangents to ensure you are in the right quadrant. In FORTRAN, you don’t have to worry about this if you use the \(\text{ATAN2}\) function instead of simply the \(\text{ATAN}\) function. The \(\text{ATAN2}\) function requires two arguments (see Table 6.1).

A sample session with a program that (hopefully) satisfies this assignment is shown on the attached page with user input shown in bold.
This program reads a real number and prints it back out. The program also remembers the "state" it is in and changes state if there was an error reading the input. The total number of states is specified by the variable "numsts".

program shell
implicit none

real a
integer state, numsts, num

a = variable that is set by read operation
state = current "state" or mode of the program
numsts = the total number of possible states (the state numbers go from zero to numsts-1)
num = status of the read operation stored here

state = 0
numsts = 5

10 read(5,*),iostat=num) a
  if (num .eq. 0) then
    print *, 'my state is ',state,' and you entered ', a
  elseif (num .lt. 0) then
    stop
  else
    state = mod(state+1,numsts)
    print *, 'I''ve changed state to state ',state
  endif
  goto 10

stop
end
Sample session of complete program with user input shown in bold.

Enter two vectors. Cartesian coordinates are assumed. Enter a non-numeric character for spherical input.

1 2 3 3 0 4
Cartesian coordinates:
Vec1: ( 1., 2., 3.)
Vec2: ( 3., 0., 4.)
Spherical coordinates:
Vec1: ( 3.7416575, 63.4349632, 36.6992302)
Vec2: ( 5., 0., 36.8699036)
Dot product: 15.
Cross product: ( 8., 5., -6.)

Spherical coordinates are assumed (angles in degrees). Enter a non-numeric character for Cartesian input.

10 45 90 20 0 0
Cartesian coordinates:
Vec1: ( 7.07106876, 7.07106638, 3.13916462E-06)
Vec2: ( 0., 0., 20.)
Spherical coordinates:
Vec1: ( 10., 45., 90.)
Vec2: ( 20., 0., 0.)
Dot product: 6.27832924E-05
Cross product: ( 141.421326, -141.421371, 0.)

5.43 20.5 17 6.0 45.0 45.0
Cartesian coordinates:
Vec1: ( 1.48704016, 0.555981398, 5.19273472)
Vec2: ( 3., 2.99999905, 4.24264145)
Spherical coordinates:
Vec1: ( 5.42999983, 20.5, 17.)
Vec2: ( 6., 45., 45.)
Dot product: 28.1599751
Cross product: ( -13.2193699, 9.26922607, 2.79317498)