## *FORTAN program to find vertical distance in time t.*

```
program vertical


   implicit none
   real,parameter :: g=9.81
   real :: s,t,u
   print *, "Enter the Value of ""t"" & ""u"""
   read *, t,u
   s=u*t-g*(t**2)*0.5
   print *, "displacement:" , s



end program Vertical
```

## FORTAN program to multiply complex parameter in a given coordinates points.

```
program complex1


   implicit none
   complex,parameter :: i=(1,1)
   complex :: x,y
   x=(10,20)
   y=(20,-10)
   print *, (i*x*y)

end program complex1
```

## *FORTAN program to find odd and even number in a given interval.*

```
program OddEven
  implicit none
  integer :: i,n2
  print *, "Enter the value of n2"
  read *,n2
  print *,"the odd integer b/w 1 &" ,n2, "are:"
  do i=1,n2,2
  print *, i
  end do
  print *,"The given integer b/w 1 &",n2, "are:"
  do i=2,n2,2
  print *,i
  end do
end program OddEven
```

## *FORTAN program to find various parameter in a projectile motion.*

```
program projectile
   implicit none
   real,parameter :: g=9.8
   real, parameter :: pi=3.1415927
   real :: a,t,u,x,y,v,vx,vy
   print *, "Enter the value of a,t,u"
   read *, a,t,u
   a=a*pi/180
   x=u*cos(a)*t
   y=u*sin(a)*t-0.5*g*(t**2)
   vx=u*cos(a)
   vy=u*sin(a)-g*t
   v=sqrt(vx*vx+vy*vy)
   print *, "x:",x,"y:",y
   print *,"v:",v
end program projectile
```

## FORTAN program to find root of given quadratic equation

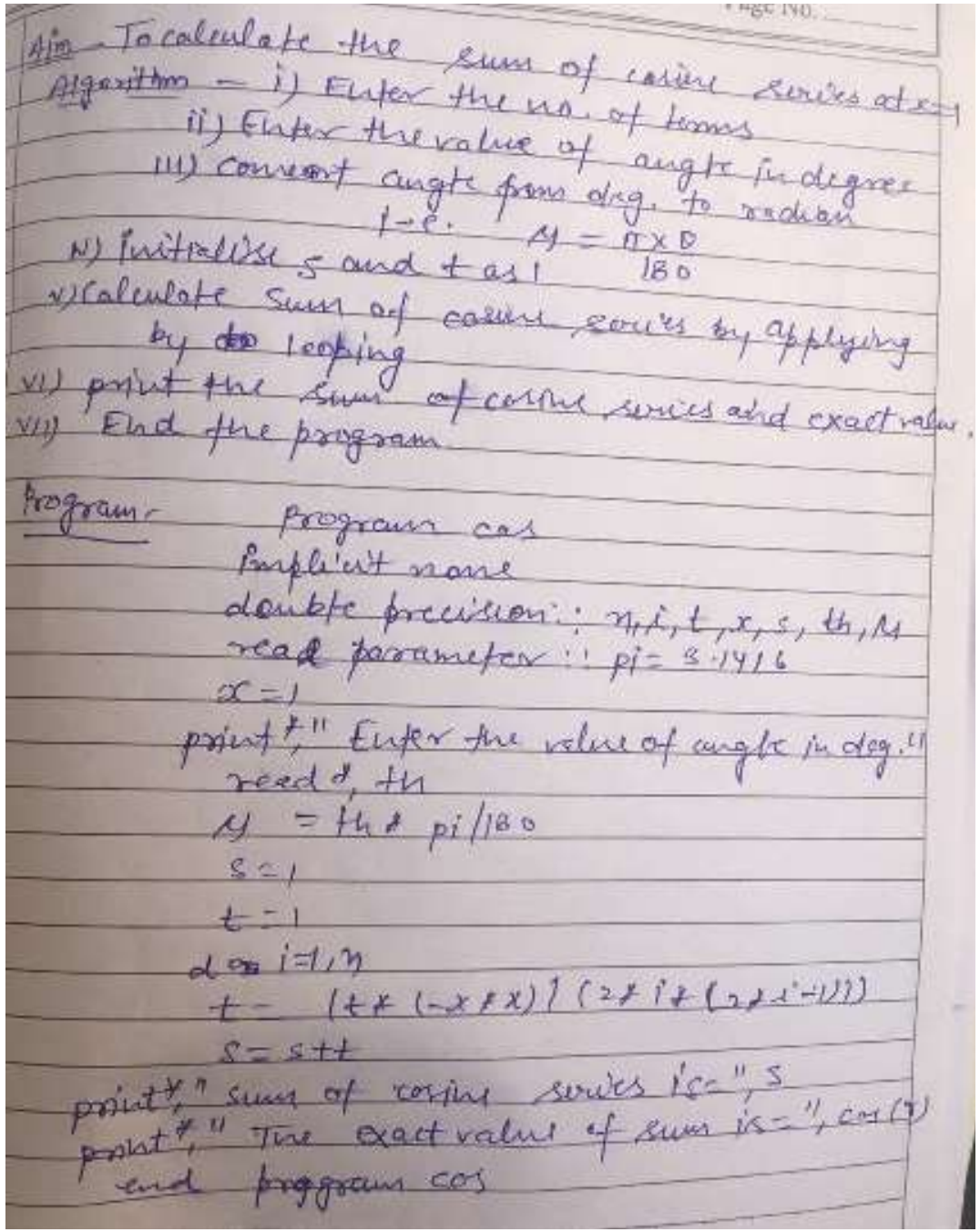
```
program quadratic
  implicit none
  real :: a,b,c,d,r1,r2
  print *, "Enter the value of a,b,c"
  read *, a,b,c
  d=b**2-4.0*a*c
  if(a/=0.0)then
      if(d==0)then
      r1=-b/(2.0*a)
      print *, "The equation has one root:",r1
  end if
  else
      print *, "Invalid input data"
      end if
      if(a/=0.0)then
          if(d>0)then
              r1=(-b+sqrt(d))/(2.0*a)
              r2=(-b-sqrt(d))/(2.0*a)
              print *, "The equation has two real root:"
              print *, r1,r2
      end if
      else
          print *, "Invalid input data"
          end if
          if(a/=0.0)then
              if(d<0)then
                  r1=-b/(2.0*a)
                  r2=sqrt(abs(d))/(2.0*a)
                  print *, "The equation has complex root:"
                  print *, "root 1=",r1,"+i",r2
                  print *, "root 2=",r1,"-i",r2
          end if
          else
              print *, "Invalid input data"
              end if
end program quadratic
```

FORTAN program to find the mean, variance and standard deviation.

```
PROGRAM MeanVariance
  IMPLICIT NONE
  INTEGER, PARAMETER :: MAX_SIZE = 50
  REAL, DIMENSION(1:MAX_SIZE) :: Data
  real:: mean,variance,stdev
  integer :: n,i
  print *, "enter the number of data elements:"

  read *,  n
  print *, "Enter the data:"
  read *,  (Data(i), i = 1, n)
  print *,  "Input Data is:"
  print *,  (Data(i), i = 1, n)
  Mean = 0.0
  DO i = 1, n
     Mean = Mean + Data(i)
  END DO
  Mean = Mean / n
  Variance = 0.0
  DO i = 1, n
     Variance = Variance + (Data(i) - Mean)**2
  END DO
  Variance = Variance / (n - 1)
  Stdev = SQRT(Variance)

  print *,  "Mean                : ", Mean
  print *,  "Variance            : ", Variance
  print *,  "Standard Deviation : ", stdev
  DO i=1,n
  END DO
  END PROGRAM MeanVariance
```

## FORTAN program to find the product of matrices.

```
program productofmatrix
    implicit none
    integer :: i,j,k,n
    real ::tmp
    real, dimension(:,:),allocatable :: a,b,c
    print*, "Enter the value of n:"
    read *, n
    allocate (a(n,n),b(n,n),c(n,n))
    print *, "Enter the elements of matrix A:"
    do j = 1,n
        do i = 1,n
            read *,a(i,j)
        end do
    end do
    print *, "Enter the elements of matrix B:"
    do j = 1,n
        do i = 1,n
            read *, b(i,j)
        end do
    end do
    do j = 1,n
        do i = 1,n
            tmp = 0.0
            do k = 1,n
                tmp = tmp+a(i,k)*b(n,j)

            end do
            c(i,j)= tmp
        end do
    end do
    print *, "The product of A and B is:"
    do j =1,n
        do i = 1,n
            print *, c(i,j)
        end do
    end do
end program productofmatrix
```



Aim - To calculate the sum of cosine series of x

Algorithm - i) Enter the no. of terms
ii) Enter the value of angle in degree
iii) Convert angle from deg. to radian
i.e. $A = \frac{\pi \times D}{180}$
iv) Initialise s and t as 1
v) Calculate sum of cosine series by applying looping
vi) print the sum of cosine series and exact value
vii) End the program

Program -

```
program cos
Implicit none
double precision :: n,i,t,x,s,th,A
read parameter : pi= 3.1416
x=1
print*," Enter the value of angle in deg."
read*, th
A = th * pi/180
s=1
t=1
do i=1,n
t= t*(-x*x)/(2*i*(2*i-1))
s=s+t
print*," sum of cosine series is=", s
print*," The exact value of sum is=", cos(x)
end program cos
```

1. Do it same for sine series
2. Write program for exponential series

Page No.

Aim - To convert cartesian coordinates to polar

Algorithm - i) Enter value of x coordinate i.e cartesian
ii) Enter the value of y-coordinate in cartesian form from i.e. x
iii) Calculate its polar coordinates i.e. $r = \sqrt{x^2+y^2}$, $\theta = \tan^{-1}(\frac{y}{x})$
iv) print polar coordinates
v) end the program

Program -

```
program polar
implicit none
real :: x, y, r, th
print*, " Enter the x-coordinates in
        cartesian form: "
read *, x
print*, " Enter the y-coordinates in cartesian form: "
read *, y
r = sqrt (x**2 + y**2)
th = atan (y/x)
print *, " The polar coordinates are: "
print *, "r =", r, "theta =", th
end program
```

Aim - To print Fibonacci series

Algorithm - i) Enter the no. of terms. n
ii) Enter the first two terms.
iii) Print first 2 terms.
iv) Initialise count = 1
v) looping for count n=2
vi) add previous two terms and move next
vii) print - fibonacci series
viii) end program.

program -
```
program Fibonacci
implicit none
INTEGER :: num, no, n1, n, count
print*," Enter the number of terms:"
read*, num
print*," Enter the first 2 terms"
read*, no, n1
print*," The first two terms are =", no, n1
count = 1
print*, "The Fibonacci series is:"
do while (count < num-2)
    n = no + n1
    print*, n
    no = n1
    n1 = n
Count = count + 1
end do
end program Fibonacci
```