```fortran
!Program is to calculate the postion of a wave in space and time using 1-D wave equation
!u_tt = A^2 u_xx, where u(x,t) is a function of x and t that tells the dispacement of wave
at postion x and time t
!The program calculating the wave genrated by a tightly streched string at both ends of
length L
!Also it is assumed that at time t = 0 the wave is at equilibrium
program wave
        real,parameter :: PI = 3.14159
        real :: string_length, time_end, x_step, t_step, phase_vel ,lambda
        real :: wave_vector, ang_vel

        !Here string_length is the ending point of string i.e. the length of string
        !time_end is the final time
        !x_step is the step size of x
        !t_step is the step size of t
        !phase_vel is the constant in the 1D wave equation

        integer :: nt, nx
        !nt and nx are the number of steps in t and x respectively

        real, dimension(:,:), allocatable :: y
        !y will be containing the the displacement of the x at time t

        real, dimension(:,:), allocatable :: ya
        !ya is the analytic solution: ya(x,t) = sin(kx)cos(wt)

        integer :: i, j
        !just two variable to move through the matrix

        write(*, '("The following program is to calculate the position of a wave genrated
by a streched string is space-time")')
        write(*, '("Using 1-D Wave Equation", /, 15X, "u_tt = phase_vel^2 u_xx", /, /)')
        write(*, '("Enter the length of the string : ")', advance = "no")
        read *, string_length
        write(*, '("Enter the final time t : ")', advance = "no")
        read *, time_end
        write(*, '("Enter the step size in x : ")', advance = "no")
        read *,x_step
        write(*, '("Enter the step size in t : ")', advance = "no")
        read *, t_step
        write(*, '("Enter the constant A : ")', advance = "no")
        read *, phase_vel

        ! Calculate the number of time and distance points
        nt = time_end/t_step + 0.5
        nx = string_length/x_step + 0.5
        print *, "nx = ", nx, ", nt = ", nt

        ! Wave vector and angular velocity
        ! We are assuming the wave is the fundamental so the wavelenght is 2L , k =
2pi/lambda , 2L/n = lambda , k = nPI/L :
        wave_vector = 7*PI/string_length
        ang_vel = phase_vel*wave_vector
        print *, "wave vector = ", wave_vector, ", angular velocity = ",ang_vel

        !Make the sure the value is 1 or less than 1 otherwise it will shoot to infinity
        lambda = phase_vel*phase_vel*t_step*t_step/(x_step*x_step)
        print *,"Lambda = ", lambda

        ! This does the analytic calculation
        allocate(ya(0:nx, 0:nt))
        do i = 0, nx
            do j = 0, nt
               ya(i, j) = sin(wave_vector*i*x_step)*cos(ang_vel*j*t_step)
            end do
        end do

        ! Write the analytic data
```

```
        open(10, file = "wave_a.txt")
        do j = 0, nt
            do i = 0, nx
                write(10,'(F0.4, 1X, F0.4, 1x, F0.4)') x_step*i, ya(i,j)
            end do
            write(10, '(/)')
        end do
        close(10)

        ! Now the numerical calculation
        !the 1 is added to both nt and nx because the initial points are also needed to be
mapped
        !as the string is streched from the both ends and initially the string is at
equilibrium thus u(0,t) = u(L,t) = 0
        !this is boundry value
        allocate(y(0:nx, 0:nt))

        do j = 0, nt
            y(0,j) = 0
            y(nx, j) = 0
        end do

        !set the 1st initial values: y(x,0) = sin(kx)
        do i = 1, nx - 1
            y(i,0) = sin(wave_vector*i*x_step)
        end do

        !set the 2nd initial values: y(x,t) = sin(wt + kx)
        do i = 1, nx-1
            y(i,1) = sin(wave_vector*i*x_step)*cos(ang_vel*t_step)
        end do

        !calculating the value of u(x,t)
        do j = 1, nt-1
            do i = 1, nx-1
              y(i,j+1) = 2*y(i,j) +  lambda*(y(i+1,j)-2*y(i,j) + y(i-1,j)) - y(i,j-1)
            end do
        end do

        open(19, file = "wave.txt")
        do j = 0, nt
            do i = 0, nx
                write(19,'(F0.4, 1X, F0.4, 1x, F0.4)') x_step*i, y(i,j)
            end do
            write(19, '(/)')
        end do

        ! All finished so free the arrays we allocated
        deallocate(ya)
        deallocate(y)

    end program wave
```