Threaded Programming coursework I: source code

```
program loops

  use omp_lib

  implicit none
  integer, parameter :: N=729
  integer, parameter :: reps=1000

  real(kind=8), allocatable ::  a(:,:), b(:,:), c(:)
  integer :: jmax(N)


  real(kind=8) :: start1,start2,end1,end2
  integer :: r

  allocate(a(N,N), b(N,N), c(N))

  call init1()

  start1 = omp_get_wtime()

  do r = 1,reps
     call loop1()
  end do

  end1  = omp_get_wtime()

  call valid1();

  print *, "Total time for ",reps," reps of loop 1 = ", end1-start1

  call init2()
  start2 = omp_get_wtime()

  do r = 1,reps
     call loop2()
  end do

  end2  = omp_get_wtime()

  call valid2();

  print *, "Total time for ",reps," reps of loop 2 = ", end2-start2


contains

subroutine init1()

  implicit none

  integer ::  i,j

  do i = 1,N
     do j = 1,N
        a(j,i) = 0.0
        b(j,i) = 3.142*(i+j)
     end do
  end do

end subroutine init1


subroutine init2()

  implicit none

  integer ::  i,j,expr

  do i = 1,N
     expr = mod(i,3*(i/30)+1)
     if (expr == 0) then
        jmax(i) = N
```

```
      else
         jmax(i) = 1
      end if
      c(i) = 0.0
   end do

   do i = 1,N
      do j = 1,N
         b(j,i) = dble(i*j+1)/dble(N*N)
      end do
   end do

end subroutine init2


subroutine loop1()

   implicit none

   integer ::  i,j

   !$omp parallel do default(none) private(j) shared(a, b) &
   !$omp                 schedule(dynamic, 16)
   do i = 1,N
      do j = N,i,-1
         a(j,i) = a(j,i) + cos(b(j,i))
      end do
   end do
   !$omp end parallel do

end subroutine loop1



subroutine loop2()

   implicit none

   integer :: i,j,k
   real (kind=8) :: rN2
```

```
  rN2 = 1.0 / dble (N*N)

  !$omp parallel do default(none) private(j, k) &
  !$omp              shared(c, b, jmax, rN2) &
  !$omp              schedule(dynamic, 8)
  do i = 1,N
     do j = 1, jmax(i)
        do k = 1,j
           c(i) = c(i) + k * log(b(j,i)) *rN2
        end do
     end do
  end do
  !$omp end parallel do

end subroutine loop2



subroutine valid1()

  implicit none

  integer :: i,j
  real (kind=8) :: suma

  suma= 0.0

  do i = 1,N
     do j = 1,N
        suma = suma + a(j,i)
     end do
  end do

  print *, "Loop 1 check: Sum of a is ", suma

end subroutine valid1



subroutine valid2()
```

```
    implicit none

    integer i
    real (kind=8) sumc

    sumc= 0.0
    do i = 1,N
       sumc = sumc + c(i)
    end do

    print *, "Loop 2 check: Sum of c is ", sumc

end subroutine valid2


end program loops
```