```
********************************************************************************************
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
********************************************************************************************
********************************************************************************************
********************************************************************************************
```
DATE: 14-02-2025 13:45:08   USER: FRTRN90   JOB: BNCHMRK   PAGE: 0000

```
BNCHMRK    FRTRN90     FILE NAME/TYPE= TESTIN/TEST4.F90
BNCHMRK    FRTRN90
BNCHMRK    FRTRN90     CREATION DATE/TIME= 14-02-2025 13:45:08
BNCHMRK    FRTRN90
BNCHMRK    FRTRN90     FILE= 001   PAGES= 0002   LINES= 000081
BNCHMRK    FRTRN90
BNCHMRK    FRTRN90     SYSTEM= LINUX(6.11.2-AMD64)
BNCHMRK    FRTRN90
BNCHMRK    FRTRN90     SYSID= ACID    SYSUSER= ACID
BNCHMRK    FRTRN90
BNCHMRK    FRTRN90     SYSUSER= ACID
BNCHMRK    FRTRN90
BNCHMRK    FRTRN90     FORM= STANDARD
BNCHMRK    FRTRN90
BNCHMRK    FRTRN90     CHAR= FONTMONO
```

```
            FFFFFFFFF RRRRRRRR  TTTTTTTT  RRRRRRRR N      NN  9999999     00000
            FFFFFFFFF RRRRRRRRR TTTTTTTT  RRRRRRRRR NN     NN 999999999   0000000
            FF        RR    RR     TT     RR     RR NNN    NN 99     99 00     00
            FF        RR    RR     TT     RR     RR NNNN   NN 99     99 00     00
            FFFFFFFFF RRRRRRRRR    TT     RRRRRRRRR NN NN  NN 999999999 00     00
            FFFFFFFFF RRRRRRRR     TT     RRRRRRRR  NN  NN NN  99999999 00     00
            FF        RR  RR       TT     RR  RR    NN   NNNN        99 00     00
            FF        RR   RR      TT     RR   RR   NN    NNN        99 00     00
            FF        RR    RR     TT     RR     RR NN     NN  99999999 0000000
            FF        RR     RR    TT     RR     RR NN      N  9999999     00000

            BBBBBBBB  N      NN  CCCCCCC  HH      HH M       M RRRRRRRR  KKKKKKKKK
            BBBBBBBBB NN     NN CCCCCCCCC HH      HH MM     MM RRRRRRRRR KKKKKKKKK
            BB     BB NNN    NN CC     CC HH      HH MMM   MMM RR    RR  KKKKKKKKK
            BB     BB NNNN   NN CC        HH      HH MMMM MMMM RR    RR  KKKKKKKKK
            BBBBBBBB  NN NN  NN CC        HHHHHHHHHH MM MMM MM RRRRRRRRR KKKKKKKKK
            BBBBBBBB  NN  NN NN CC        HHHHHHHHHH MM  M  MM RRRRRRRR  KKKKKKKKK
            BB     BB NN   NNNN CC        HH      HH MM     MM RR  RR    KKKKKKKKK
            BB     BB NN    NNN CC     CC HH      HH MM     MM RR   RR   KKKKKKKKK
            BBBBBBBBB NN     NN CCCCCCCCC HH      HH MM     MM RR    RR  KKKKKKKKK
            BBBBBBBB  NN      N  CCCCCCC  HH      HH MM     MM RR    RR  KKKKKKKKK

                          00000     00000         1
                         0000000   0000000        11
                        00    00 00     00       111
                        00    00 00     00        11
                        00    00 00     00        11
                        00    00 00     00        11
                        00    00 00     00        11
                        00    00 00     00        11
                         0000000   0000000      111111
                          00000     00000       111111
```

```fortran
program lid_driven_cavity
   implicit none
   integer, parameter :: N = 50    ! grid size (NxN grid)
   real :: dx, dy, dt, Re        ! grid spacing, time step, Reynolds number
   real :: u(N, N), v(N, N), p(N, N)  ! velocity and pressure fields
   integer :: i, j, step
   real :: start_time, end_time, elapsed_time

   ! Parameters
   dx = 1.0 / (N-1)    ! Grid spacing in x direction
   dy = 1.0 / (N-1)    ! Grid spacing in y direction
   dt = 0.001          ! Time step size
   Re = 100            ! Reynolds number

   ! Initialize arrays
   u = 0.0
   v = 0.0
   p = 0.0

   ! Initialize the top boundary (lid) velocity
   u(N, :) = 1.0

   ! Start timing
   call cpu_time(start_time)

   ! Main loop for time stepping
   do step = 1, 1000
       call compute_velocity(u, v, p, dx, dy, dt, Re)
       call update_pressure(p, dx, dy)

       ! Output or check convergence
       if (mod(step, 100) == 0) then
           print *, 'Step: ', step
       end if
   end do

   ! Stop timing
   call cpu_time(end_time)
   elapsed_time = end_time - start_time
   print *, 'Elapsed time for CFD simulation: ', elapsed_time, ' seconds'

contains

   ! Function to update the velocity and pressure fields (simplified)
   subroutine compute_velocity(u, v, p, dx, dy, dt, Re)
       real, dimension(:,:), intent(inout) :: u, v, p
       real, intent(in) :: dx, dy, dt, Re
       integer :: i, j

       ! Simple explicit method for velocity (simplified)
       do i = 2, N-1
           do j = 2, N-1
               u(i, j) = u(i, j) - dt * ( (u(i, j) * (u(i+1, j) - u(i-1, j))) / (2*dx) + &
                         (v(i, j) * (u(i, j+1) - u(i, j-1))) / (2*dy) )
           end do
       end do

       ! Simple velocity update for v (similar)
       do i = 2, N-1
           do j = 2, N-1
```

```fortran
          v(i, j) = v(i, j) - dt * ( (u(i, j) * (v(i+1, j) - v(i-1, j))) / (2*dx) + &
                                     (v(i, j) * (v(i, j+1) - v(i, j-1))) / (2*dy) )
      end do
   end do
 end subroutine compute_velocity

 ! Function to solve for pressure (simplified Poisson equation solver)
 subroutine update_pressure(p, dx, dy)
   real, dimension(:,:), intent(inout) :: p
   real, intent(in) :: dx, dy
   integer :: i, j

   ! Simple pressure Poisson equation (Jacobi iteration)
   do i = 2, N-1
      do j = 2, N-1
         p(i, j) = 0.25 * ( p(i+1, j) + p(i-1, j) + p(i, j+1) + p(i, j-1) )
      end do
   end do
 end subroutine update_pressure

end program lid_driven_cavity
```