```
module general                  !General Module to be used everywhere
    real,allocatable::x(:)      !list position of n particles
    integer::n                  !no. of particles
    real::xcut,l                !cut of potential, length of box
end module

program LJ_fluid
    use general
    implicit none
    integer::i,j,k,nmc,m
    real::dx, r, r1, r2, du, del, xtest, w
    real::T, a, u, uold, unew, kb, utot, s

!kb = boltzman constant, T = temperature, a = particle diameter
!nmc = no. of MC cycles, m = cycles to be excluded

    n = 20;     l = 40;     nmc = 5000; kb = 1;     s = 0
    T = 0.3;    xcut = 3;   a = 0.95;   del = 0.3;  m = 3000
    open(1,file="result.dat")

    allocate(x(n))              !Allocating Latice points
    do i = 1,n                  !Initializing Latice
        x(i) = (i-1)*a
    enddo

    do i = 1, nmc
        do k = 1,n
            call random_number(r1)
            call random_number(r2)
            j = int(n*r1) + 1   !selecting random latice point to change distance
            dx = del*(2*r2 - 1) !random change in distance
            xtest = x(j) + dx
            if (xtest > l) then !Boundary condition upper bound
                xtest = xtest - l
            elseif (xtest < 0) then !Boundary condition lower bound
                xtest = xtest + l
            endif
            call energy(j,x(j),u); uold = u     !Calculating energy of x(j)
            call energy(j,xtest,u); unew = u    !Calculating energy of x(j) + dx
            if (unew < uold) then               !Accepting condition
                x(j) = xtest
            else
                du = unew - uold                !Metropolis condition
                w = exp(-du/(kb*T))
                call random_number(r)
                if (r <= w) then
                    x(j) = xtest
                endif
            endif
        enddo
        call totalenergy(utot)  !Calculating energy with new latice distances
        write(1,*) utot

        if (i > m) then                         !Excluding m cycles
            s = s + utot
        endif
    enddo
    write(*,*)"Average Value of Energy is:", s/(nmc-m)
end program

! LJ Potential [V = 4*epsilon*((sigma/r)^12 - (sigma/r)^6)] for every particle

! Calculates energy of j^th particle w.r.t. all other particles
subroutine energy(j,xj,u)
    use general
    implicit none
```

```
67      integer::i,j
68      real::d,u,xj
69      u = 0.0
70      do i = 1,n
71          if(i == j) cycle
72          d = xj - x(i)
73          if(d > l/2.0) then      !Boundary condition upper bound
74              d = d - l
75          elseif(d < -l/2.0) then !Boundary condition lower bound
76              d = d + l
77          endif
78
79          if(abs(d) < xcut) then  !Boundary condition w.r.t. cut off potential
80              u = u + 4 * ( (1/d**12) - (1/d**6) )
81          endif
82      enddo
83      return
84  end subroutine
85
86  !Calculate energy of all particle w.r.t. all other particles
87  subroutine totalenergy(u)
88      use general
89      implicit none
90      integer::i,j
91      real::d,u
92      u = 0.0
93      do i = 1,n-1
94          do j = i+1, n
95              d = x(i) - x(j)
96              if(d > l/2.0) then      !Boundary condition upper bound
97                  d = d - l
98              elseif(d < -l/2.0) then !Boundary condition lower bound
99                  d = d + l
100             endif
101
102             if(abs(d) < xcut) then  !Boundary condition w.r.t. cut off potential
103                 u = u + 4 * ( (1/d**12) - (1/d**6) )
104             endif
105         enddo
106     enddo
107     return
108 end subroutine
109
110 !OUTPUT
111 !Average Value of Energy is:  -13.2113447 (a = 0.95)
112 !Average Value of Energy is:  -14.7483358 (a = 1.12
```