```
module general              !General Module to be used everywhere
    integer,allocatable::x(:)   !list of n particles
    integer::n          !no. of particles
end module

program Ising_Model
    use general
    implicit none
    integer::nmc, i, k, a
    real::J, B, kb, T, r, uold, unew, du, w, utot
    J = 1.0;     B = 0.0;   kb = 1.0;   T = 0.7
    nmc = 500000;    n = 1000
    allocate(x(n))
    do i = 1,n
        x(i) = 1
    enddo
    open(1, file = "result.dat");   open(2, file = "States.dat")
    write(2,*)"Initial State"
    write(2,*)x
    call total_energy(J,B,utot)
!Energy,    Energy per Spin,     Magnetization,   Magnetization per Spin
    write(1,*) utot, utot/n, real(sum(x)), real(sum(x))/(n*1.0)

    do i = 1,nmc
        do k = 1,n

            call random_number(r)
            a = int(r*n) + 1
            if (a > n) then
                a = n
            endif

            call energy(J,B,a,uold)
            x(a) = -x(a)
            call energy(J,B,a,unew)

            if(unew > uold) then
                du = unew - uold
                w = exp(-du/(kb*T))
                call random_number(r)
                if(r >= w) then
                    x(a) = -x(a)
                endif
            endif
        enddo

        call total_energy(J,B,utot)
        write(1,*) utot, utot/n, real(sum(x)), real(sum(x))/(n*1.0)
    enddo

        write(2,*)"Final State"
    write(2,*)x

end program

! E = -J * Sum(Si*Sj) - B * Sum(Si)

subroutine energy(j,b,i,u)  !Calculate total energy of given State
    use general
    implicit none
    integer::i
    real::d,u,j,b,e
    u = 0.0
    if(i == 1)then
        e = -j*x(i)*(x(n) + x(i-1)) - b*x(i); u = u + e
    elseif(i == n) then
```

```
67          e = -j*x(i)*(x(1) + x(i-1)) - b*x(i); u = u + e
68      else
69          e = -j*x(i)*(x(i-1) + x(i+1)) - b*x(i); u = u + e
70      endif
71      return
72  end subroutine
73
74  subroutine total_energy(j,b,u)
75      use general
76      implicit none
77      integer::i
78      real::d,u,j,b,e
79      u = 0.0
80      do i = 1,n
81          if(i == n) then
82              e = -j*(x(n)*x(1)); u = u + e
83          else
84              e = -j*(x(i)*x(i+1)); u = u + e
85          endif
86          u = u - b*sum(x)
87      enddo
88      return
89  end subroutine
```