```
Program Melting
IMPLICIT NONE
INTEGER Imax,Jmax,i,j,k,Ntime,Nf,w1,w2,w3 !nf=number of fin
REAL Tm,tw !Tm=melting temperature=east wall temperature, uo=initial velocity
REAL rwf,lf,rHf  !rwf= relative width of fin lf=relative length of fin,
      !rHf=relative position of fin on west wall from bottom
REAL dx,dy,dt,delta,deltat,Kr,teta,tetam,tetastep
REAL visco, pr ,ra ,ste, omega, omegat, omegat1, rhoo, alpha, alphat , gbeta
REAL rcf,Ang,savn
parameter(Imax=50,Jmax=50) ! number of grids in x and y
parameter(pr=1,ste=10.0D0,visco=0.1D0) ! Pr, Ra and Stefan
Parameter(Tm=0.0,tw=1.0D0) ! melting temperature and wall temperature
parameter(Kr=10.0D0) ! the ratio of thermal conductivity of fin
REAL f(0:8,0:Imax,0:Jmax)
REAL feq(0:8,0:Imax,0:Jmax),rho(0:Imax,0:Jmax)
REAL u(0:Imax,0:Jmax),v(0:Imax,0:Jmax),flo(0:Imax,0:Jmax),flj(0:Imax,0:Jmax)
REAL g(0:4,0:Imax,0:Jmax),geq(0:4,0:Imax,0:Jmax),th(0:3*Imax,0:3*Jmax)
REAL w(0:8),ww(0:4),cx(0:8),cy(0:8)
WRITE(*,*) "IN THE NAME OF ALLAH"
WRITE(*,*) "THIS CODE SOLVE MELTING WITH LBM"
WRITE(*,*) "INPUT inclination angle"
READ(*,*) Ang ! the inclination angle
WRITE(*,*) "INPUT Ra number"
READ(*,*) Ra
WRITE(*,*) "with fin==1 *** without fin==0"
READ(*,*) Nf  !the number of fin
IF(Nf.eq.1)THEN
      WRITE(*,*) "INPUT WIDTH OF FIN (ratio)"
      READ(*,*) rwf
      WRITE(*,*) "LENGTH OF FIN"
      READ(*,*) rHf ! the fin length ratio=(Hf/Imax)
      WRITE(*,*) "INPUT center point of fin from bottom"
      READ(*,*) rcf ! the position of fin on right wall of cavity
      w1 = Jmax * (rcf - rwf / 2.)   !the position of fin base
      w2 = w1 + rwf * (Jmax)         !position of another edge of fin base
      w3 = rHf * (Jmax)              !position of fin tip
      PRINT*,w1,w2,w3
ENDIF
OPEN(2,file='uvtcontour.plt')
OPEN(6,file='nuav0.txt')
OPEN(7,file='nu.txt')
OPEN(50,file='Sav.txt')
cx(:)  = (/0.0D0,1.0D0,0.0D0,-1.0D0,0.0D0,1.0D0,-1.0D0,-1.0D0,1.0D0/)
cy(:)  = (/0.0D0,0.0D0,1.0D0,0.0D0,-1.0D0,1.0D0,1.0D0,-1.0D0,-1.0D0/)
w(:)   = (/4.D0/9.D0,1.D0/9.D0,1.D0/9.D0,1.D0/9.D0,1.D0/9.D0,1.D0/36.D0,1.D0/36.D0,1.D0/36.D0,1.D0/36.D0/)
ww(:)  = (/1.D0/3.D0,1.D0/6.D0,1.D0/6.D0,1.D0/6.D0,1.D0/6.D0/)
rhoo   = 1.0D0
dx     = 1.0D0
```

```
dy     = dx
dt     = 1.0D0
deltaT = tw - Tm      !Maximum Temperature difference
alpha  = visco / pr
gbeta  = ra * visco * alpha / (FLOAT(Jmax * Jmax * Jmax))  !g*B
omega  = 1 / (3.0D0 * visco + 0.5D0)       !omega for velocity in Fluid
omegat = 1 / (3.0D0 * alpha + 0.5D0)      !omega for Temperature in Fluid
omegat1= 1 / (3.0D0 * alpha * Kr + 0.5D0)  ! omega for fin
DO j=0,Jmax
          DO i=0,Imax
            rho(i,j)= rhoo
            th(i,j) = Tm
            u(i,j)  = 0.0D0
            v(i,j)  = 0.0D0
            flo(i,j)= Tm
          ENDDO
ENDDO
tetam   = 0D0
tetastep= 0.1D0
44 Ntime = Ntime + 1 !Ntime=time counter
teta = alpha * ste * Ntime / FLOAT(Imax * Jmax) !teta=St*FO,FO=time*alpha/(H*L)
CALL Collesion(Imax,Jmax,rho,u,v,cx,cy,w,omega,gbeta, Ang,th,feq,f)
CALL Streaming(Imax,Jmax,f)
CALL Bouncebanck(Imax,Jmax,w1,w2,w3,Nf,f)
CALL Rhouv(Imax,Jmax,cx,cy,u,v,th,w1,w2,w3,Nf,rho,f)
CALL Collt(Imax,Jmax,cx,cy,u,v,th,tw,Tm,ste,ww,omegat, omegat1,w1,w2,w3,Nf,&
          flo,flj,geq,g)
CALL StreamingT(Imax,Jmax,g)
CALL Gbound(g,tw,w,Imax,Jmax)
CALL Tcalcu(g,th,Imax,Jmax)

PRINT*,Ntime," *** ",th(Imax/2,Jmax/2)," *** ",u(Imax/2,Jmax/2)," *** "
IF (Ntime.eq.100) THEN ! FOR WRITE  RESULT AFTER 100 TIMESTEP
          CALL RESULT(tw,u,v,th,Imax,Jmax,w1,w2,w3,Nf,teta)
          CALL Meltfront(th,Imax,Jmax,w1,w2,w3,savn,Nf,teta)
ENDIF

IF(mod(teta,tetastep).lt.0.001D0.and.int(10*teta).ne. tetam) then !for saving output file in tetastep by 0.1
          tetam = tetam + 1
          CALL RESULT(tw,u,v,th,Imax,Jmax,w1,w2,w3,Nf,teta)
          CALL Meltfront(th,Imax,Jmax,w1,w2,w3,savn,Nf,teta)
END IF
IF(savn.lt.1)THEN
          GOTO 44
ENDIF

STOP

END
SUBROUTINE Collesion(Imax,Jmax,rho,u,v,cx,cy,w,omega,gbeta, Ang,th,feq,f)

IMPLICIT NONE
```

```
INTENT(in) ::Imax,Jmax,rho,u,v,cx,cy,w,omega,gbeta,Ang,th
INTENT(out)::f

INTEGER I,J,K,Imax,Jmax
REAL    omega,tref,gbeta,Ang,t1,t2,force

REAL f(0:8,0:Imax,0:Jmax),th(0:3*Imax,0:3*Jmax)
REAL feq(0:8,0:Imax,0:Jmax),rho(0:Imax,0:Jmax)
REAL w(0:8),cx(0:8),cy(0:8)
REAL u(0:Imax,0:Jmax),v(0:Imax,0:Jmax)
tref = 0.5D0
DO i=0,Imax
        DO j=0,Jmax
        t1 = u(i,j) * u(i,j) + v(i,j) * v(i,j)
          DO k=0,8
          t2=u(i,j)*cx(k)+v(i,j)*cy(k)
          force=3.D0 * w(k) * gbeta * cosd(Ang) * (th(i,j)-tref) * cy(k) * rho(i,j)+&
                3.D0 * w(k) * gbeta * sind(Ang) * (th(i,j)-tref) * cx(k) * rho(i,j)
          feq(k,i,j)=rho(i,j) * w(k) * (1.0D0 + 3.0D0 * t2 + 4.5D0 * t2 * t2 - 1.50D0 * t1)
                  IF (th(i,j).gt.0.001) THEN
f(k,i,j)=omega*feq(k,i,j)+(1.D0-omega)*f(k,i,j)+force
                  ELSE
                    f(k,i,j)=omega*feq(k,i,j)+(1.D0-omega)*f(k,i,j)
                  END IF
        ENDDO
   ENDDO
ENDDO
RETURN
END
SUBROUTINE Streaming(Imax,Jmax,f)
IMPLICIT NONE
INTENT(in)   ::Imax, Jmax
INTENT(inout)::f
INTEGER I,J,Imax,Jmax
REAL f(0:8,0:Imax,0:Jmax)
DO j=0,Jmax
        DO i=Imax,1,-1 ! RIGHT TO LEFT
          f(1,i,j)=f(1,i-1,j)
        ENDDO
        DO i=0,Imax-1 ! LEFT TO RIGHT
          f(3,i,j)=f(3,i+1,j)
        ENDDO
ENDDO
DO j=Jmax,1,-1 ! TOP TO BOTTOM
        DO i=0,Imax
          f(2,i,j)=f(2,i,j-1)
        ENDDO
        DO i=Imax,1,-1
          f(5,i,j)=f(5,i-1,j-1)
        ENDDO
        DO i=0,Imax-1
          f(6,i,j)=f(6,i+1,j-1)
        ENDDO
ENDDO
DO j=0,Jmax-1 !BOTTOM TO TOP
        DO i=0,Imax
          f(4,i,j)=f(4,i,j+1)
        ENDDO

        DO i=0,Imax-1
```

```
            f(7,i,j)=f(7,i+1,j+1)
          ENDDO

          DO i=Imax,1,-1
            f(8,i,j)=f(8,i-1,j+1)
          ENDDO
ENDDO
RETURN
END
 SUBROUTINE Bouncebanck(Imax,Jmax,w1,w2,w3,Nf,f)
IMPLICIT NONE
INTENT(in)   ::Imax,Jmax,w1,w2,w3,Nf
INTENT(inout)::f
INTEGER w1,w2,w3,nf,i,j,Imax,Jmax
REAL f(0:8,0:Imax,0:Jmax)
 DO j=0,Jmax
! Bouncebanck:part_1 For West Boundary(NO SLIP)
   f(1,0,j)   = f(3,0,j)
   f(5,0,j)   = f(7,0,j)
   f(8,0,j)   = f(6,0,j)
! Bouncebanck:part_2 For East Boundary (NO SLIP)
   f(3,Imax,j)= f(1,Imax,j)
   f(7,Imax,j)= f(5,Imax,j)
   f(6,Imax,j)= f(8,Imax,j)
ENDDO
DO i=0,Imax
! Bouncebanck:part_3 For South Boundary (NO SLIP)
   f(2,i,0)   = f(4,i,0)
   f(5,i,0)   = f(7,i,0)
   f(6,i,0)   = f(8,i,0)
```

```
! Bouncebanck:part_4 For North Boundary (NO SLIP)
   f(4,i,Jmax)= f(2,i,Jmax)
   f(8,i,Jmax)= f(6,i,Jmax)
   f(7,i,Jmax)= f(5,i,Jmax)
ENDDO
!Bouncebanck:part_5 bounce back for fin
IF(Nf.eq.1)THEN
          DO i=0,w3
!Bouncebanck:part_5_1 For TOP surface
            f(2,i,w2) = f(4,i,w2)
            f(5,i,w2) = f(7,i,w2)
            f(6,i,w2) = f(8,i,w2)
!Bouncebanck:part_5_2 For BOTTOM
            f(4,i,w1) = f(2,i,w1)
            f(8,i,w1) = f(6,i,w1)
            f(7,i,w1) = f(5,i,w1)
          ENDDO
!Bouncebanck:part_5_3 For right surface (SIDE)
          DO j=w1,w2
            f(1,w3,j) = f(3,w3,j)
            f(5,w3,j) = f(7,w3,j)
            f(8,w3,j) = f(6,w3,j)
          ENDDO
ENDIF
RETURN
 END
 SUBROUTINE Rhouv(Imax,Jmax,cx,cy,u,v,th,w1,w2,w3,Nf,rho,f)
IMPLICIT NONE
INTENT(in) ::Imax,Jmax,th,w1,w2,w3,Nf,f,cx,cy
INTENT(out)::rho,u,v
INTEGER I,J,k,Imax,Jmax,w1,w2,w3,Nf
```

```
REAL    usum,vsum,ssum
REAL
f(0:8,0:Imax,0:Jmax),rho(0:Imax,0:Jmax),u(0:Imax,
0:Jmax),&

v(0:Imax,0:Jmax),cx(0:8),cy(0:8),th(0:3*Imax,0:3*J
max)
!Rhouv:part_1 calculation of density
DO j=0,Jmax
        DO i=0,Imax
          ssum=0.0D0
          DO k=0,8
                ssum = ssum + f(k,i,j)
          ENDDO
          rho(i,j)= ssum
        ENDDO
ENDDO
!Rhouv:part_2 calculation of X- AND Y-
VELOCITY
DO i=0,Imax
        DO j=0,Jmax
          usum = 0.0D0
          vsum = 0.0D0
!Rhouv:part_2_1
          DO k=0,8
                usum = usum + f(k,i,j) * cx(k)
                vsum = vsum + f(k,i,j) * cy(k)
          ENDDO
!IT ASSUMED VELOCITY FOR T LOWER
THAN 0.001 IS ASSUMED ZERO( IN
UNMELTED ZONE)
          IF (th(i,j).gt.0.001D0) THEN
                u(i,j) = usum / rho(i,j)
                v(i,j) = vsum / rho(i,j)
          ELSE
                u(i,j) = 0.0D0
                v(i,j) = 0.0D0
          ENDIF
        ENDDO
ENDDO
DO i=0,Imax
  u(i,0)    = 0.0D0
  v(i,0)    = 0.0D0
  u(i,Jmax) = 0.0D0
  v(i,Jmax) = 0.0D0
ENDDO
DO j=0,Jmax
  u(0,j)    = 0.0D0
  v(0,j)    = 0.0D0
  u(Imax,j) = 0.0D0
  v(Imax,j) = 0.0D0
ENDDO
IF (Nf.eq.1) THEN
        DO j=w1,w2
          DO i=0,w3
                u(i,j) = 0.0D0
                v(i,j) = 0.0D0
          ENDDO
        ENDDO
ELSE
ENDIF
RETURN
 END
 SUBROUTINE
Collt(Imax,Jmax,cx,cy,u,v,th,tw,Tm,ste,ww,omegat,
omegat1,w1,w2,w3,&
          Nf,flo,flj,geq,g)
Implicit none
```

```
INTENT(in) ::w1,w2,w3,u,v,cx,cy,Imax,Jmax,ww, omegat1,omegat,tw,Tm,th
INTENT(inout)::g
INTEGER I,J,k,Imax,Jmax,w1,w2,w3,Nf
REAL    ste,deltaT, omegat,omegat1,tm,tw
REAL g(0:4,0:Imax,0:Jmax),geq(0:4,0:Imax,0:Jmax),th(0:3*Imax,0:3*Jmax),&
    flo(0:Imax,0:Jmax)
REAL w(0:8),cx(0:8),cy(0:8),flj(0:Imax,0:Jmax)
REAL u(0:Imax,0:Jmax),v(0:Imax,0:Jmax),ww(0:4)
deltaT = tw - Tm

!Collt:part1 calculating liquid fraction
DO i=0,Imax
        DO j=0,Jmax
          IF (th(i,j).gt.0.001D0) THEN
                flj(i,j)=th(i,j)
          ELSE
                flj(i,j)=0.0D0
          ENDIF
        ENDDO
ENDDO
!Collt:part2 Calculating the collision
DO i=0,Imax
        DO j=0,Jmax
          IF (Nf.eq.1.and.j.ge.w1 .and. j.le.w2 .and. i.le.w3) THEN ! for fin
                DO k=0,4
                        geq(k,i,j)= th(i,j) * ww(k) * (1.0D0 + 3.0D0 * (u(i,j) * cx(k) &
                                + v(i,j) * cy(k)))
                        g(k,i,j)  = omegat1 * geq(k,i,j) + (1.0D0 - omegat1) * g(k,i,j)
                ENDDO
        ELSEIF(th(i,j).gt.0.001) THEN ! for liquid zone
                DO k=0,4
                        geq(k,i,j)= th(i,j) * ww(k) * (1.0D0 + 3.0D0 * (u(i,j) * cx(k) &
                                + v(i,j) * cy(k)))
                        g(k,i,j)  = omegat  * geq(k,i,j) + (1.0D0 - omegat) * g(k,i,j) &
                                - (ww(k) * deltaT / ste * (flj(i,j) - flo(i,j)))
                ENDDO
        ELSE ! for solid zone
                DO k=0,4
                        g(k,i,j) = 0.0
                ENDDO
        END IF
        ENDDO
ENDDO

!Collt:part3 Replacing the old liquid fraction with new value in each node
DO j=0,Jmax
        DO i=0,Imax
          flo(i,j) = flj(i,j)
        ENDDO
ENDDO

RETURN

END
SUBROUTINE StreamingT(Imax,Jmax,g)

IMPLICIT NONE
```

```
INTENT(in)   ::Imax,Jmax
INTENT(inout)::g

INTEGER I,J,Imax,Jmax
REAL    g(0:4,0:Imax,0:Jmax)
DO j=0,Jmax
        DO i=Imax,1,-1 ! RIGHT TO LEFT
          g(1,i,j) = g(1,i-1,j)
        ENDDO
        DO i=0,Imax-1 ! LEFT TO RIGHT
          g(3,i,j) = g(3,i+1,j)
        ENDDO
ENDDO
DO j=Jmax,1,-1 ! TOP TO BOTTOM
        DO i=0,Imax
          g(2,i,j) = g(2,i,j-1)
        ENDDO
ENDDO
DO j=0,Jmax-1 !BOTTOM TO TOP
        DO i=0,Imax
          g(4,i,j) = g(4,i,j+1)
        ENDDO
ENDDO
 RETURN
END
SUBROUTINE Gbound(g,tw,w,Imax,Jmax)
Implicit none
INTENT(in)   ::tw,w,Imax,Jmax
INTENT(inout)::g
INTEGER I,J,Imax,Jmax
Real    tw
REAL g(0:4,0:Imax,0:Jmax)
REAL w(0:8)
 DO j=0,Jmax
   g(1,0,j) = tw - (g(0,0,j) + g(2,0,j) + g(3,0,j) +
g(4,0,j))
ENDDO
!Gbound:part_2 East Boundary Condition, adiabatic
 DO j=0,Jmax
   g(4,Imax,j) = g(4,Imax-1,j)
   g(3,Imax,j) = g(3,Imax-1,j)
   g(2,Imax,j) = g(2,Imax-1,j)
   g(1,Imax,j) = g(1,Imax-1,j)
   g(0,Imax,j) = g(0,Imax-1,j)
ENDDO
! Gbound:part_3 Top Boundary Condition,
Adiabatic
DO i=0,Imax
   g(4,i,Jmax) = g(4,i,Jmax-1)
   g(3,i,Jmax) = g(3,i,Jmax-1)
   g(2,i,Jmax) = g(2,i,Jmax-1)
   g(1,i,Jmax) = g(1,i,Jmax-1)
   g(0,i,Jmax) = g(0,i,Jmax-1)
ENDDO
! Gbound:part_4 bottom Boundary Condition,
Adiabatic
DO i=0,Imax
   g(4,i,0)    = g(4,i,1)
   g(3,i,0)    = g(3,i,1)
   g(2,i,0)    = g(2,i,1)
   g(1,i,0)    = g(1,i,1)
   g(0,i,0)    = g(0,i,1)
ENDDO
RETURN
END
```

```
SUBROUTINE Tcalcu(g,th,Imax,Jmax)
IMPLICIT NONE
INTENT(in) ::Imax,Jmax,g
INTENT(out)::th
INTEGER I,J,K,IMAX,JMAX
REAL ssumt
REAL g(0:4,0:Imax,0:Jmax),th(0:3*Imax,0:3*Jmax)
DO j=0,Jmax
        DO i=0,Imax
          ssumt = 0.0D0 ! a dummy argument for summution
          DO k=0,4
                ssumt = ssumt + g(k,i,j)
          ENDDO
          th(i,j) = ssumt
        ENDDO
ENDDO
RETURN
END
SUBROUTINE RESULT(tw,u,v,th,Imax,Jmax,w1,w2,w3,Nf,teta)
Implicit none
INTENT(in)   ::Imax,Jmax,w1,w2,w3,Nf,teta,tw
INTENT(inout)::th
INTENT(out)  ::u,v
INTEGER I,J,Imax,Jmax,w1,w2,w3,Nf
REAL nu1,nu2,nu3,nu4,nu5,snu,Nuav,r1,teta,yy,tw,nu
REAL u(0:Imax,0:Jmax),v(0:Imax,0:Jmax),th(0:3*Imax,0:3*Jmax)
DO i=0,Imax
        DO j=0,Jmax
          IF (th(i,j).lt.0.001D0) THEN
                u(i,j) = 0.0D0
                v(i,j) = 0.0D0
          END IF
        ENDDO
ENDDO
!Result:part_1 monitoring mesh and U,V,T(melt fraction) contour

WRITE(2,*)"VARIABLES=X,Y,U,V,T"
WRITE(2,*)"ZONE ","I=",Imax+1,"J=",Jmax+1,",","F=BLOCK"
DO j=0,Jmax
  WRITE(2,*)(i / FLOAT(Imax),i=0,Imax)
ENDDO
DO j=0,Jmax
  WRITE(2,*)(j / FLOAT(Jmax),i=0,Imax)
ENDDO
DO j=0,Jmax
  WRITE(2,*)(u(i,j),i=0,Imax)
ENDDO
DO j=0,Jmax
  WRITE(2,*)(v(i,j),i=0,Imax)
ENDDO
DO j=0,Jmax
  WRITE(2,*)(th(i,j),i=0,Imax)
ENDDO
!Result:part_2 calculating Nusselt number
yy  = yy + 1.
snu = 0.0
IF(Nf.eq.1)THEN
        snu = 0.0
        r1  = -1.
        DO j=1,w1-1
```

```
            nu1 = ((3.0 * th(0,j) - 4.0 * th(1,j) +
th(2,j)) / 2.0) * FLOAT(Jmax)!w1
            snu = snu + ABS(nu1)
            r1  = r1 + 1.
            WRITE(7,*) r1,nu1
          ENDDO
          DO i=0,w3-1
            nu2 = ((3.0 * th(i,w1) - 4.0 * th(i,w1-1.) +
th(i,w1-2.0)) / 2.0) &
                    * FLOAT(Jmax)
            snu = snu + ABS(nu2)
            r1  = r1 + 1.
            WRITE(7,*) r1,nu2
          ENDDO
          DO j=w1+2,w2-2
            r1  = r1 + 1.
            nu3 = ((3.0 * th(w3,j) - 4.0 * th(w3+1.,j) +
th(w3+2.,j)) / 2.0) &
                    * FLOAT(Jmax)
            snu = snu + ABS(nu3)
            WRITE(7,*) r1,nu3
          ENDDO
          DO i=w3,1,-1
            nu4 = ((3.0 * th(i,w2) - 4.0 * th(i,w2+1.) +
th(i,w2+2.0)) / 2.0) &
                    * FLOAT(Jmax)
       nu = snu + ABS(nu4)
            WRITE(7,*) r1,nu4
            r1 = r1 + 1.
            WRITE(7,*) r1,nu4
          ENDDO
          DO j=w2+1,Jmax-1
            nu5 = ((3.0 * th(0,j) - 4.0 * th(1,j) +
th(2,j)) / 2.0) * FLOAT(Jmax)
      snu = snu + ABS(nu5)
            r1  = r1 + 1.
            WRITE(7,*) r1,nu5
          ENDDO
          Nuav = snu / (FLOAT(Jmax))
          WRITE(6,*)yy,Nuav
 ELSE
          r1  = -1
          snu = 0.0
          DO j=0,Jmax
            nu = ((3.0 * th(0,j) - 4.0 * th(1,j) + th(2,j))
/ 2.0) * FLOAT(Jmax)
            snu= snu + ABS(nu)
            r1 = r1 + 1.
            WRITE(7,*) r1,snu
          ENDDO
          Nuav = snu / (FLOAT(Jmax))
          WRITE(6,*)teta,Nuav
ENDIF
RETURN
END
 SUBROUTINE
Meltfront(th,Imax,Jmax,w1,w2,w3,savn,Nf,teta)
IMPLICIT NONE
INTENT(in) ::Imax,Jmax,teta,Nf,w1,w2,w3,th
INTENT(out)::Savn
INTEGER I,J,Imax,Jmax,w1,w2,w3,Nf
REAL a,b,teta,ss,sss,Savn,Sav,tt
REAL th(0:3*Imax,0:3*Jmax),s(0:Jmax)
a   = 0.0
ss  = 0.0
sss = 0.0
Sav = 0.0
```

```
DO i=0,Imax
  b = -1.
11  b = b + 1.
      IF(b.lt.Jmax)THEN
        GOTO 22
      ELSE
        GOTO 55
      ENDIF
22    IF(th(i,b).ge.0.001)THEN
        a = a + 1.
        GOTO 11
      ELSE
        GOTO 12
      ENDIF
12    b = b + 1
      IF(b.lt.Jmax) THEN
        GOTO 23
      ELSE
        GOTO 55
      ENDIF
23    IF(th(i,b).le.0.001)THEN
        GOTO 12
      ELSE
        a = a + 1.
        GOTO 22
      ENDIF
55  ENDDO
! for melting over fin surface
! this section calculate the area in fin that melting front reachs there.
 IF(Nf.eq.1) THEN
      tt = (w2 - w1)
      DO j=w1,w2,tt
        DO i=0,w3
          IF(th(i,j).gt.0.001) THEN
            s(j)=i
          ENDIF
        ENDDO
      ENDDO
      DO j=w1,w2,tt
        ss = ss + s(j)
      ENDDO
      sss = ss * (w2 - w1) / 2.
      Savn= (a - sss) / (FLOAT(jmax * jmax) - (w2 - w1) * w3)
 ELSE
   Savn=a/FLOAT(Jmax*Jmax)
 ENDIF

 WRITE(50,*)teta, Savn
RETURN
END
```