```fortran
  Program DataRed
    use Crystallographic_Symmetry, only: Space_Group_Type, Set_SpaceGroup,
Write_SpaceGroup
    use String_Utilities,          only: u_case
    use Reflections_Utilities,     only: hkl_absent, hkl_equiv, hkl_s
    use Math_gen,                  only: sort, asind
    use Crystal_types,             only: Crystal_Cell_Type, Set_Crystal_Cell,
Write_Crystal_Cell

    implicit none

    integer, parameter           :: nref=400000, inp=1, ihkl=3, irej=4, iin=10,
i_scr=99
    integer, dimension(3,nref)   :: h
    integer, dimension(3)        :: h1,h2
    real,    dimension(nref)     :: intens, sigma, twtheta, intav, sigmav
    integer, dimension(nref)     :: itreat, iord, nequv, ini, fin,  warn
    real,    dimension(48)       :: weight
    character(len=256)           :: filein, fileout, filecon
    character(len=132)           :: line, cmdline, title
    character(len=20)            :: spg_symb
    character(len=6)             :: key
    real, dimension(3)           :: cel,ang
    type (Space_Group_Type)      :: grp_espacial
    type (Crystal_Cell_Type)     :: celda
    character(len=*),parameter,dimension(0:1) :: warn_mess=(/"
    ",  &
                                                              " <- Dubious
reflection"/)
    Logical :: Friedel=.true.,  cell_given=.false., wave_given=.false.
    real    :: sig, suma, suman,  Rint,   &
               wavel,sigg, delt, warning, t_start, t_end
    integer :: i,j,k, ier, nr=0,  ns, rej,  len_cmdline, &
               lenf, nin, cent
    integer :: iargc, narg


  !-----------------------------  Treating  the command line
      narg=iargc()
      len_cmdline=0
      if(narg > 0) then
              call getarg(1,cmdline)
              len_cmdline=len_trim(cmdline)
      end if

    if(len_cmdline /= 0) then
        lenf=index(cmdline," ")-1
        filecon=cmdline(1:lenf)//".red"
        open(unit=iin,file=filecon,status="old",iostat=ier,action="read")
        if(ier/=0) then
          write(unit=*,fmt="(3a)") " => File: ", trim(filecon)," not found!"
          stop
        end if
        read(unit=iin,fmt="(a)") title
    else
        write(unit=*,fmt="(a)") " => Please invoke the program as: 'data_red
myfile' where myfile.res is the input file"
        stop
    end if
  !----------------------------- End  Treating  the command line

    call cpu_time(t_start)
    write(unit=*,fmt="(a)") "       ==============================="
    write(unit=*,fmt="(a)") "       DATA REDUCTION PROGRAM: DataRed"
    write(unit=*,fmt="(a)") "       ==============================="
    write(unit=*,fmt="(a)") "       "
    twtheta(:) =0.0
```

```
!------------------------------ Start reading the input command file

  read(unit=iin,fmt="(a,a)") key, filein
  filein=adjustl(filein)
  write(unit=*,fmt="(a,a)") " => Name of the input file: ", trim(filein)
  read(unit=iin,fmt="(a,a)") key, fileout
  fileout=adjustl(fileout)
  write(unit=*,fmt="(a,a)") " => Code of the output file: ", trim(fileout)
  warning=0.30  ! 30% error for warning equivalent reflections

     do
        read(unit=iin,fmt="(a)", iostat=ier) line
        if(ier /= 0) exit
        line=adjustl(line)
        if(line(1:1) == "!") cycle

        key=u_case(line(1:5))

        Select Case(key(1:5))

           Case("SPGR ")
             spg_symb=adjustl(line(6:))

           Case("NFRDL")
              Friedel=.false.

           Case("CELL ")
              read(unit=line(7:),fmt=*)  cel, ang
              call Set_Crystal_Cell(cel,ang,Celda)
              cell_given=.true.

           Case("WAVE ")
              read(unit=line(7:),fmt=*)  wavel
              wave_given=.true.

        End Select

     end do


  ! check that all is O.K.
  if(.not. cell_given) then
    write(unit=*,fmt=*)" => UNIT CELL not GIVEN! Modify your input file."
    stop
  end if
  if(.not. wave_given) then
    write(unit=*,fmt=*)" => WAVELENGTH not GIVEN! Modify your input file."
    stop
  end if

!------------------------------ End reading the input command file


!------------------------------ Start reading the INTENSITY input file
  open(unit=inp, file=filein, status="old", iostat=ier,action="read")
  nr=0

 !Reading reflections and calculate 2theta

  do
   nr=nr+1
   read(unit=inp,fmt=*,iostat=ier) h1(:), intens(nr), sigma(nr)
    if(ier /= 0) then
     nr=nr-1
     exit
    end if
```

```
      twtheta(nr)=2.0* ASIND( hkl_s(h1,celda)*WAVEL)
      if(twtheta(nr) < 0.0001) ier=1
      h(:,nr)=h1(:)
      if(sigma (nr) <= 0.0 )  sigma(nr)=0.004
    end do

    write(unit=*,fmt="(a,i6)") " => Total number of reflections read: ", nr

  !----------------------------- End  reading the INTENSITY input file

!
!  Order the reflections by ascending twtheta
!
    call sort(twtheta,nr,iord)

   ! Non-elegant way of ordering things
    open(unit=i_scr,status="scratch",form="unformatted",action="readwrite")
    do i=1,nr
      k=iord(i)
      write(unit=i_scr) h(:,k), intens(k), sigma(k), twtheta(k)
    end do
    rewind (unit=i_scr)
    do k=1,nr
      read(unit=i_scr) h(:,k), intens(k), sigma(k), twtheta(k)
    end do
    close(unit=i_scr)
    write(unit=*,fmt="(a)")" => Reflections ordered by ascending two-theta
O.K.!"
!
!  Set symmetry
!
    call Set_SpaceGroup(spg_symb,grp_espacial)

!
!  Opening file for rejected reflections
!
    open(unit=irej, file=trim(fileout)//".rej", status="replace",action="write")
   write(unit=irej,fmt="(a)") "         REJECTED REFLECTIONS (Symmetry
forbidden)"
   write(unit=irej,fmt="(a)") "   h   k   l   Intensity       Sigma  TwoTheta
  I/sig"
   write(unit=irej,fmt="(a)") "
=====================================================
!
!  First loop over reflections
!
   nin=0
   itreat(:)=0
   ini(:)=0
   fin(:)=0

   rej=0

   do i=1,nr                   !Loop over all measured reflections

     if(itreat(i) == 0) then   !If not yet treated do the following
       h1(:)=h(:,i)
       if(hkl_absent(h1,grp_espacial)) then  !reject absent reflections
         rej=rej+1
          write(unit=irej,fmt="(3i4,2f12.3,f10.4,f10.2)") h1(:),intens(i),
sigma(i),twtheta(i), intens(i)/sigma(i)
         cycle
       end if

       nin=nin+1    !update the number of independent reflections
       itreat(i)=i  !Make this reflection treated
       sig =1.0/sigma(i)**2
       ini(nin)=i   !put pointers for initial and final equivalent reflections
       fin(nin)=i
       nequv(nin)=1 !One reflection for the moment equivalent to itself

         do j=i+1,nr       !look for equivalent reflections to the current (i)
in the list
           if(abs(twtheta(i)-twtheta(j)) > 0.001) exit
            h2=h(:,j)
              if(hkl_equiv(h1,h2,grp_espacial,Friedel)) then ! if h1 eqv h2
               itreat(j) = i                                 ! add h2 to the
list equivalent to i
               nequv(nin)=nequv(nin)+1                       ! update the
number of equivalents
               sig=sig + 1.0/sigma(j)**2
               fin(nin)=j
              end if
         end do

         ns=0
         do j=ini(nin),fin(nin)
           if(itreat(j) == i) then
             ns=ns+1
             weight(ns)=(1.0/sigma(j)**2)/sig
           end if
         end do

         suma=0.0
         ns=0
         do j=ini(nin),fin(nin)
           if(itreat(j) == i) then
             ns=ns+1
             suma=suma+weight(ns)*intens(j)
           end if
         end do

         intav(nin)=suma

         suma=0.0
         ns=0
         do j=ini(nin),fin(nin)
           if(itreat(j) == i) then
             ns=ns+1
             delt= intav(nin)-intens(j)
             if(abs(delt)/intav(nin) > warning) warn(nin)=1
             suma=suma+weight(ns)*delt*delt
           end if
         end do

         sigmav(nin)=sqrt(suma)
         sigg=SUM(sigma(ini(nin):fin(nin)))/max(1.0,real(fin(nin)-ini(nin)))
         if(sigmav(nin) < sigg) sigmav(nin) = sigg

     end if !itreat
   end do
!
! Second loop over reflections to calculate R-int
!
   ns=0
   suma =0.0
   suman=0.0
   do i=1,nin
     k=ini(i)
     if(nequv(i) < 2 ) cycle
     sig=0.0
     do j=ini(i),fin(i)
       if(itreat(j) == k) then
         sig=sig+1.0/sigma(j)**2
       end if
     end do
     sig=1.0/sig
     do j=ini(i),fin(i)
       if(itreat(j) == k) then
         ns=ns+1
         suma=suma+abs(intav(i)-intens(j))
         suman=suman+intens(j)
       end if
     end do
   end do
   Rint = 100.0*suma/max(1.0,suman)
!
!---  Writing the list of rejected and merged reflections
!

    open(unit=ihkl, file=trim(fileout)//".int", status="replace",action="write")
   write(unit=ihkl,fmt="(a)") title
   write(unit=ihkl,fmt="(a)") "(3i4,2F12.3,i5,4f8.2)"
   write(unit=ihkl,fmt="(f9.5,a)") wavel,"   0   0"
   cent=0
   do i=1,nin
     j=ini(i)
     h1(:)=  h(:,j)
     if(hkl_equiv(h1,-h1,grp_espacial,.false.)) then
        cent=cent+1   !calculate the number of acentric reflections
        write(unit=ihkl,fmt="(3i4,2f12.3,i5,4f8.2,a)") h1(:),intav(i),sigmav(i),
1,0.0,0.0,0.0,0.0, warn_mess(warn(i))//" Centric"
     else
        write(unit=ihkl,fmt="(3i4,2f12.3,i5,4f8.2,a)") h1(:),intav(i),sigmav(i),
1,0.0,0.0,0.0,0.0, warn_mess(warn(i))
     end if
   end do

  !--------- All calculations have been done!

   write(unit=*,fmt="(/,a,i6)")" => Number of reflections read               :
", nr
   write(unit=*,fmt="(a,i6)")  " => Number of valid independent   reflections:
", nin
   write(unit=*,fmt="(a,i6)")  " => Number of Centric            reflections:
", cent
   write(unit=*,fmt="(a,i6)")  " => Number of rejected (absences) reflections:
", rej
   write(unit=*,fmt="(a,f6.2)")" => R-internal for equivalent reflections (%):
", Rint

   write(unit=*,fmt="(a)")     " => Program finished O.K.!, look in output
files!"
   write(unit=*,fmt="(a,a)")   "       General    Output    file: ",
trim(fileout)//".out"
   write(unit=*,fmt="(a,a)")   "    Independent  reflections file: ",
trim(fileout)//".int"
   write(unit=*,fmt="(a,a)")   "    Rejected     reflections file: ",
trim(fileout)//".rej"
   write(unit=*,fmt="(a )")    "          "
   call cpu_time(t_end)
   write(unit=*,fmt="(a,f10.2,a)")  "  CPU-Time: ", t_end-t_start," seconds"
   stop
  End Program DataRed
```