```
-- ----------------------------------------------------------------------
-- Copyright (c) 1990-2010 Oracle Corp.  All rights reserved.
-- ----------------------------------------------------------------------

REMOVE PROGRAM DBMS_RR, DBMS_RR, TRUE, FORCE;

CREATE OR REPLACE
PROGRAM DBMS_RR
AUTHID CURRENT_USER
TRUSTED
PURITY
DEFAULT
PARALLEL_ENABLE
TIME COMPILE
TIME PARSE
SYSTEM
-- BEGIN DBMS_RR.SQL(10)(0)
-- NAME DBMS_RR - DATA REPLAY PACKAGE FOR CAPTURING
-- DESCRIPTION DBMS_RR PROVIDES THREE MAJOR SETS WHICH CONSTITUTE DATA REPLAY:
-- 1. CAPTURE - DBMS_RR.START CAPTURES DATA FROM
-- 2. REPLAY - DBMS_RR.START REPLAYS DATA
-- 3. REPORT - DBMS_RR.REPORT_DIVERGENCE
-- MODIFIED
-- 08/20/09 - DBMS_RR(10): REPLAYS
-- 08/20/09 - DBMS_RR(10): REPLAYS
-- 08/20/09 - DBMS_RR(10): CAPTURE
-- 08/20/09 - DBMS_RR(10): CAPTURE
-- END DBMS_RR.SQL(10)(0)
-- BEGIN DBMS_RR.PKS(10)(0)
-- NAME DBMS_RR - DATA REPLAY PACKAGE
-- END DBMS_RR.PKS(10)(0)
AS
-- BEGIN DBMS_RR.PKS(10)(0)
-- PUBLIC CONSTANTS
-- ----------------
-- ----------------
-- ----------------
```

```
     1      *PSIF(I,4))/DSQRT(1.0D0*B(7))
        IF (7 .LE. NBAS) PSIF(I,7) = ((X(I)-A(7))*PSIF(I,6)-DSQRT(1.0D0*B(7))
     1      *PSIF(I,5))/DSQRT(1.0D0*B(8))
        IF (8 .LE. NBAS) PSIF(I,8) = ((X(I)-A(8))*PSIF(I,7)-DSQRT(1.0D0*B(8))
     1      *PSIF(I,6))/DSQRT(1.0D0*B(9))
        IF (9 .LE. NBAS) PSIF(I,9) = ((X(I)-A(9))*PSIF(I,8)-DSQRT(1.0D0*B(9))
     1      *PSIF(I,7))/DSQRT(1.0D0*B(10))
        ENDDO
C
        RETURN
        END


        SUBROUTINE DGQRUL_BETA (NGAUSS,QX,QW)
C
C       CALCULATE GAUSS QUADRATURE RULE FOR GUMBEL  PDF (WEIGHT)
C
C
C

C   -------------  JUST FOR ALFA = 5, BETA = 5;  0<= X <=1 --------
C   ------ OTHER CASE NEED USING STIELTJES PROCESS RECALCULATE A AND B


      IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 B(10), A(10), QX(50),QW(50)
C




      IF (NGAUSS .EQ. 1) THEN
        QX(1) =  0.5
        QW(1) =  1.0D0
      ELSEIF (NGAUSS .EQ. 2) THEN
        QX(1) =  3.4924432771E-001
        QW(1) =  5.0000000000E-001
        QX(2) =  6.5075567229E-001
        QW(2) =  5.0000000000E-001
      ELSEIF (NGAUSS .EQ. 3) THEN
        QX(1) =  2.5980776929E-001
        QW(1) =  1.9696969697E-001
        QX(2) =  5.0000000000E-001
        QW(2) =  6.0606060606E-001
        QX(3) =  7.4019223071E-001
        QW(3) =  1.9696969697E-001
      ELSEIF (NGAUSS .EQ. 4) THEN
        QX(1) =  2.0129742715E-001
        QW(1) =  7.6169789309E-002
        QX(2) =  3.9618876278E-001
        QW(2) =  4.2383021069E-001
        QX(3) =  6.0381123722E-001
        QW(3) =  4.2383021069E-001
        QX(4) =  7.9870257285E-001
        QW(4) =  7.6169789309E-002

      ELSEIF (NGAUSS .EQ. 5) THEN
        QX(1) =  1.6070082773E-001
        QW(1) =  3.0562430134E-002
        QX(2) =  3.2129662783E-001
        QW(2) =  2.4566134609E-001
        QX(3) =  5.0000000000E-001
        QW(3) =  4.4755244755E-001
        QX(4) =  6.7870337217E-001
        QW(4) =  2.4566134609E-001
        QX(5) =  8.3929917227E-001
        QW(5) =  3.0562430134E-002

      ELSE

      PRINT *, 'NGAUSS TOO BIG OR NEGATIVE! '

      STOP




      ENDIF



C
        RETURN
        END









C-------------------------------------------------------

      subroutine gauss2(ngauss,a, b, qx, qw)
c-------------------------------------------------------
C     produce gauss quadrature rule from recursion coefficient a, b
c               by solving a eigenvalue problem
C     input:    ngauss,a, b,
C     output:   qx, qw
c
C
C-------------------------------------------------------

      USE EVCSB_INT
      USE EPISB_INT
      USE UMACH_INT
      USE WRRRN_INT

      INCLUDE "link_fnl_shared.h"


      IMPLICIT NONE
      REAL*8 b(20), a(20), qx(10),qw(10)
      INTEGER ngauss, i, j

      INTEGER LDA, LDEVEC, N, NCODA, IWK(ngauss)
      PARAMETER (NCODA=1, LDA=NCODA+1)
      INTEGER NOUT
      REAL*8 AA(LDA,ngauss), EVAL(ngauss), EVEC(ngauss,ngauss), PI,
     1 AA2(1,1), ACOPY( ngauss*NCODA),WK(ngauss)

      N = ngauss
      LDEVEC = ngauss
      qx = 0.
      qw = 0.
      if (ngauss .gt. 1) then
      do j = 2,ngauss
        AA(1,j) = dsqrt(b(j))
      enddo
      AA(1,1) = 0.

      do j = 1,ngauss
        AA(2,j) = a(j)
      enddo
C Find eigenvalues and vectors
c      CALL DE4CSB (N, AA, LDA, NCODA, EVAL, EVEC, LDEVEC, ACOPY, WK,IWK)
      CALL EVCSB (AA, NCODA, EVAL, EVEC)
C Compute performance index
      PI = EPISB(ngauss,AA,NCODA,EVAL,EVEC)

      else
        AA2(1,1) = A(1)
C Find eigenvalues and vectors
c      CALL DE4CSB (N, AA2, LDA, NCODA, EVAL, EVEC, LDEVEC,
c     1     ACOPY, WK,IWK)
      CALL EVCSB (AA2, 0, EVAL, EVEC)
C Compute performance index
      PI = EPISB(ngauss,AA2,0,EVAL,EVEC)

      endif

C Print results
      CALL UMACH (2, NOUT)
      CALL WRRRN ('EVAL', EVAL, 1, ngauss, 1)
      CALL WRRRN ('EVEC', EVEC)
      WRITE (NOUT,'(/,A,F6.3)') ' Performance index = ', PI

      do i = 1,ngauss
!        if (i .gt.10) pause
        qx(i) = EVAL(ngauss-i+1)
        qw(i) = b(1)*EVEC(1, ngauss-i+1)**2.0
      enddo


      end
```