```python
import os, sys
sys.path.append(os.path.join(os.path.dirname(__file__), '../ch02'))

from math import sqrt

from program2_1 import Dvector
from program2_2 import Dmatrix
from program2_3 import input_matrix
from program9_2 import householder

N = 4

def main():
    eps = 10.0 ** -8.0

    a = Dmatrix(1, N, 1, N) # 行列領域の確保

    # ファイルのオープン
    with open("input_eigen.dat", "r") as fin:
        with open("result_eigen.dat", "w") as fout:
            input_matrix( a, 'A', fin, fout ) # 行列 A の入出力
            a_hh = householder( a, N )               # ハウスホルダー法
            a_qr = qr( a_hh, eps, N )                   # QR法

            # 結果の出力
            print("QR法の結果は")
            for i in range(1, N+1):
                for j in range(1, N+1):
                    print("{:10.7f}\t".format(a_qr[i][j]), end="")
                print()

            print("固有値は")
            for i in range(1, N+1):
                print("{:10.7f}\t".format(a_qr[i][i]), end="")
            print()


# QR法
def qr(a: Dmatrix, eps: float, n: int) -> Dmatrix:
    # 領域の確保
    q    = Dmatrix(1, n, 1, n)
    work = Dvector(1, n)
    a_qr = a.copy()
    m = n
    while m > 1:
        # 収束判定
        if abs(a_qr[m][m-1]) < eps:
            m = m - 1
            continue

        # 原点移動
        s = a_qr[n][n]
        if m == n:  # m=n のときは原点移動なし
            s = 0.0
        for i in range(1, m+1):
            a_qr[i][i] -= s

        # QR法
        for i in range(1, m+1):
            for j in range(1, m+1):
                q[i][j] = 0.0      # Q を m x m 単位行列で初期化
            q[i][i] = 1.0

        # R と Q の計算
        for i in range(1, m):
            r = sqrt( a_qr[i][i]*a_qr[i][i] + a_qr[i+1][i]*a_qr[i+1][i] )
            if r == 0.0:
                sint = 0.0
                cost = 0.0
            else:
                sint = a_qr[i+1][i] / r
                cost = a_qr[i][i] / r
            for j in range(i+1, m+1):
                tmp = a_qr[i][j]*cost + a_qr[i+1][j]*sint
                a_qr[i+1][j] = -a_qr[i][j]*sint + a_qr[i+1][j]*cost
                a_qr[i][j] = tmp
            a_qr[i+1][i] = 0.0
            a_qr[i][i] = r
            for j in range(1, m+1):        # Q は P の転置
                tmp = q[j][i]*cost + q[j][i+1]*sint
                q[j][i+1] = -q[j][i]*sint + q[j][i+1]*cost
                q[j][i] = tmp

        # RQ の計算
        for i in range(1, m+1):
            for j in range(i, m+1):
                work[j] = a_qr[i][j]
            for j in range(1, m+1):
                a_qr[i][j] = sum( ( work[k] * q[k][j] for k in range(i, m+1) ) )

        # 原点移動後の処理
        for i in range(1, m+1):
            a_qr[i][i] += s

    return a_qr


if __name__ == "__main__":
    main()
```