```
/* Math/CS 466/666 Midterm Solutions

   Problem 2. Compute the spectral norm of the 4 × 4 matrix A with entries given by the
   formula A_ij = √(i + 2j). */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void matprint(int n,double A[n][n]){
    for(int i=0;i<n;i++) for(int j=0;j<n;j++){
        printf("%8g%c",A[i][j],j==n-1?'\n':' ');
    }
}
double vecdp(int n,double u[n],double v[n]){
    double r=0;
    for(int i=0;i<n;i++) r+=u[i]*v[i];
    return r;
}
void matmul(int n,double y[n],double A[n][n],double x[n]){
    for(int i=0;i<n;i++) y[i]=0;
    for(int i=0;i<n;i++) for(int j=0;j<n;j++) y[i]+=A[i][j]*x[j];
}

double normS(int n,double A[n][n]){
/* Compute ‖A‖ = ρ(B)^{1/2} where B = A^t A. We use the power method to find the
   spectral radius of ρ(B). Since the eigenvalues of B are positive, the power method is
   guaranteed to converges to the spectral radius for almost every choice of x_0. */
    double x[n],y[n],B[n][n];
    for(int i=0;i<n;i++) for(int j=0;j<n;j++) B[i][j]=0;
    for(int k=0;k<n;k++) for(int i=0;i<n;i++) for(int j=0;j<n;j++){
        B[i][j]+=A[k][i]*A[k][j];
    }
    /* Choose the initialization randomly so the chances of x_0 lying in a eigen-subspace
       of B is unlikely. One could call the routine multiple times to be sure. */
    for(int i=0;i<n;i++) y[i]=2.0*rand()/RAND_MAX-1.0;
    double lambdaold=0;
    printf("\n%3s %22s\n","k","lambda_k");
    for(int k=1;k<100;k++){
        double r=sqrt(vecdp(n,y,y));
        for(int i=0;i<n;i++) x[i]=y[i]/r;
        matmul(n,y,B,x);
        double lambda=vecdp(n,x,y);
        printf("%3d %22.14e\n",k,lambda);
        if(fabs(lambda-lambdaold)<=1e-14*fabs(lambda)) break;
```

```
49          lambdaold=lambda;
50      }
51      return sqrt(lambdaold);
52  }
53
54  #define N 4
55  double A[N][N];
56
57  int main(){
58      printf("Math/CS 466/666 Midterm\nProblem 2\n\n");
59      for(int i=1;i<=N;i++) for(int j=1;j<=N;j++){
60          /* Mathematical notation assumes i and j range from 1 to N. As C arrays
               always start at zero we store the matrix as A_ij = A[i-1][j-j] */
63          A[i-1][j-1]=sqrt(i+2*j);
64      }
65      printf("A=\n");
66      matprint(N,A);
67      printf("\nnormS(A)=%.14e\n",normS(N,A));
68      printf("\nnormS(A)=%.14e\n",normS(N,A));
69      return 0;
70  }
```