```
// Copyright (c) Wenjun Ying, School of Mathematical Sciences and Institute
// of Natural Sciences, Shanghai Jiao Tong University, Shanghai 200240.

// This file is free software; as a special exception the author gives
// unlimited permission to copy and/or distribute it, with or without
// modifications, as long as this notice is preserved.

// This file is distributed in the hope that it will be useful, but
// WITHOUT ANY WARRANTY, to the extent permitted by law; without even the
// implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <cmath>

#include <limits.h>

const float EPSILON = 1.0E-6;

void computeMatrixProduct(float **L, float **U, int n, float **B)
{
  for (unsigned int i = 0; i < n; i++) {
    for (unsigned int j = 0; j < n; j++) {
      float sum = 0.0;
      for (unsigned int k = 0; k < n; k++) {
        sum += L[i][k] * U[k][j];
      }
      B[i][j] = sum;
    }
  }
}

float computeMaxError(const float *x, const float *y, int n)
{
  float e = 0.0;
  for (unsigned int i = 0; i < n; i++) {
    float d = fabs(x[i] - y[i]);
    if (d > e) {
      e = d;
    }
  }
  return e;
}

void printVector(const char *info, const float *x, int n)
{
  std::cout << info << std::endl;
  for (unsigned int i = 0; i < n; i++) {
    std::cout << x[i] << " ";
  }
  std::cout << std::endl;
}

void printMatrix(const char *info, float **A, int n)
{
  for (unsigned int i = 0; i < n; i++) {
    for (unsigned int j = 0; j < n; j++) {
      if (fabs(A[i][j]) < EPSILON) {
        printf("        ");
      } else {
        printf("%7.3f ", A[i][j]);
      }
    }
    std::cout << std::endl;
  }
  std::cout << std::endl;
}
```

```
int makeLUdecomposition(float **A, int n, float **L, float **U)
{
  for (unsigned int i = 0; i < n; i++) {
    for (unsigned int j = 0; j < n; j++) {
      U[i][j] = A[i][j];
      L[i][j] = 0.0;
    }
    L[i][i] = 1.0;
  }

  for (unsigned int k = 0; k < (n - 1); k++) {
    for (unsigned int i = k + 1; i < n; i++) {
      if (fabs(U[k][k]) < EPSILON) {
        return (-1);
      }
      float lik = U[i][k] / U[k][k];
      for (unsigned int j = k; j < n; j++) {
        U[i][j] = U[i][j] - lik * U[k][j];
      }
      L[i][k] = lik;
    }
  }

  return 0;
}

int makeLUdecomposition2(float **A, int n, float **L, float **U) // by the Dooli
ttle method
{
  for (unsigned int i = 0; i < n; i++) {
    for (unsigned int j = 0; j < n; j++) {
      U[i][j] = 0.0;
      L[i][j] = 0.0;
    }
    L[i][i] = 1.0;
  }

  for (unsigned int k = 0; k < n; k++) {
    // compute the k^{th} row of the upper triangular matrix "U"
    for (unsigned int j = k; j < n; j++) {
      float sum = 0.0;
      for (unsigned int m = 0; m < k; m++) {
        sum += L[k][m] * U[m][j];
      }
      U[k][j] = A[k][j] - sum;
    }

    // compute the k^{th} column of the lower triangular matrix "L"
    for (unsigned int i = k + 1; i < n; i++) {
      float sum = 0.0;
      for (unsigned int m = 0; m < k; m++) {
        sum += L[i][m] * U[m][k];
      }
      if (fabs(U[k][k]) < EPSILON) {
        return (-1);
      }
      L[i][k] = (A[i][k] - sum) / U[k][k];
    }
  }

  return 0;
}

int solveByGaussElimination(float **A, float *b,
                            int n, float *x)
{
  for (unsigned int k = 0; k < (n - 1); k++) {
    for (unsigned int i = k + 1; i < n; i++) {
      if (fabs(A[k][k]) < EPSILON) {
        return (-1);
      }
      float lik = A[i][k] / A[k][k];
      for (unsigned int j = k; j < n; j++) {
        A[i][j] = A[i][j] - lik * A[k][j];
      }
      b[i] = b[i] - lik * b[k];
    }
  }

  printMatrix("A:", A, n);

  for (int k = (n - 1); k >= 0; k--) {
    float sum = 0.0;
    for (unsigned int j = k + 1; j < n; j++) {
      sum += A[k][j] * x[j];
    }
    x[k] = (b[k] - sum) / A[k][k];
  }

  return 0;
}

float randfloat(void)
{
  float v = (rand()%INT_MAX) / (INT_MAX - 1.0);
  return v;
}

float randfloat(float a, float b)
{
  float v = a + (b - a) * randfloat();
  return v;
}

int main(int argc, char *argv[])
{
  int n = 0;
  if (argc > 1) {
    int n0 = atoi(argv[1]);
    if (n0 > 1) {
      n = n0;
    }
  }
  if (n <= 1) {
    std::cout << "Please input the matrix/vector dimension: ";
    std::cin >> n;
    if (n <= 1) {
      std::cout << "invalid dimension found" << std::endl;
      exit(EXIT_FAILURE);
    }
  }

  float **A = new float*[n];
  for (unsigned int i = 0; i < n; i++) {
    A[i] = new float[n];
    for (unsigned int j = 0; j < n; j++) {
      A[i][j] = 0.0;
    }
  }

  //////////////////////////////////////////////////////////////////////////////

  srand(time(0));

  for (unsigned int i = 0; i < n; i++) {
    for (unsigned int j = 0; j < n; j++) {
      A[i][j] = randfloat(-1.0, 1.0);
    }
```

```
  }

  printMatrix("A:", A, n);

  /* float *z = new float[n];
  for (unsigned int i = 0; i < n; i++) {
    z[i] = randfloat(-1.0, 1.0);
  }

  float *b = new float[n];
  for (unsigned int i = 0; i < n; i++) {
    b[i] = 0.0;
    for (unsigned int j = 0; j < n; j++) {
      b[i] += A[i][j] * z[j];
    }
  }

  float *x = new float[n];
  for (unsigned int i = 0; i < n; i++) {
    x[i] = 0.0;
  }

  int status = solveByGaussElimination(A, b, n, x);
  if (status < 0) {
    std::cout << "Failed to solve the system" << std::endl;
    exit(EXIT_FAILURE);
  }

  printVector("x:", x, n);
  printVector("z:", z, n);

  float err = computeMaxError(x, z, n);
  std::cout << "max error = " << err << std::endl; */

  //////////////////////////////////////////////////////////////////////////////

  float **L = new float*[n];
  for (unsigned int i = 0; i < n; i++) {
    L[i] = new float[n];
    for (unsigned int j = 0; j < n; j++) {
      L[i][j] = 0.0;
    }
  }

  float **U = new float*[n];
  for (unsigned int i = 0; i < n; i++) {
    U[i] = new float[n];
    for (unsigned int j = 0; j < n; j++) {
      U[i][j] = 0.0;
    }
  }

  //int status = makeLUdecomposition(A, n, L, U); // by the Gauss elimination

  int status = makeLUdecomposition2(A, n, L, U); // by the Doolittle method
  if (status < 0) {
    std::cout << "Failed to make LU decomposition" << std::endl;
    exit(EXIT_FAILURE);
  }

  printMatrix("L:", L, n);
  printMatrix("U:", U, n);

  float **B = new float*[n];
  for (unsigned int i = 0; i < n; i++) {
    B[i] = new float[n];
    for (unsigned int j = 0; j < n; j++) {
      B[i][j] = 0.0;
    }
  }

  computeMatrixProduct(L, U, n, B);

  printMatrix("B:", B, n);

  // free pointers

  for (unsigned int i = 0; i < n; i++) {
    delete[] A[i];
  }
  delete[] A;

  for (unsigned int i = 0; i < n; i++) {
    delete[] L[i];
  }
  delete[] L;

  for (unsigned int i = 0; i < n; i++) {
    delete[] U[i];
  }
  delete[] U;

  for (unsigned int i = 0; i < n; i++) {
    delete[] B[i];
  }
  delete[] B;

  /* delete[] b;
  delete[] x;
  delete[] z; */

  return EXIT_SUCCESS;
}
```