TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

ĐỀ THI GIỮA HỌC KỲ VÀ BÀI LÀM

Tên học phần: Toán ứng dụng CNTT
Mã học phần: Hình thức thi: *Tự luận*
Đề số: **0002** Thời gian làm bài: 90 phút *(không kể thời gian chép/phát đề)*
Được sử dụng tài liệu khi làm bài.

Họ tên: ……Nguyễn Hữu Khoa……**Lớp**:……22T_DT5……**MSSV**:…102220237…...

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV_HọTên.pdf và nộp bài thông qua MSTeam

Câu 1 (***2 điểm***): Cho số nguyên dương N=403. Viết chương trình bằng C/C++ có sử dụng hàm thực hiện:

- Tìm số lượng các số nguyên dương (giữa 1 và N) là nguyên tố cùng nhau với N (N>1), liệt kê và tính tổng của chúng N.
- Tìm số nguyên tố gần N nhất và chia hết cho 3.

 # Trả lời: Dán code vào bên dưới:

```
#include <bits/stdc++.h>

//bai 2

using namespace std;

#define ll long long

bool is_prime(long long n) {

   if (n <= 1) {

      return false;

   }

   for (int i = 2; i * i <= n; i++) {

      if (n % i == 0) {

         return false;

      }

      }

   return true;

}

struct Element

{

   ll value;
```

```
    int count;
};
ll GCD(ll a, ll b)
{
    if (b == 0) return a;
    else return GCD(b, a % b);
}
void factorize(vector<Element> *A, ll n)
{
        int count = 0;
        ll temp = n;
            while (!(temp % 2))
            {
                temp = temp/2;
                count++;
            }

            if (count)
            {
                // cout << 2 << "^" << count;
                struct Element B;
                B.count = count;
                B.value = 2;
                A->push_back(B);
            }
            for (long long i = 3; i <= sqrt(n); i += 2)
            {
                count = 0;
                while (temp % i == 0) {
                    count++;
```

```
                temp = temp / i;
            }
            if (count){
                struct Element B;
                B.count = count;
                B.value = i;
                A->push_back(B);
                    // cout << "*" << i << "^" << count << endl;
            }

        }
}
void ETF(ll number)
{
    int S;
    cout << "list: \n";
    for (int i = 1; i<number; i++)
    {
        ll a = GCD(i,number);
        if (a == 1){
            S+=i;
            cout << i << " ";
        }
    }
    cout << "\n";
    cout << "sum:  " << S << "\n";
}
void ETFFORMULA(ll n)
{
    vector<Element> temp;
```

```
    factorize(&temp, n);
    // for (int i = 0; i < temp.size(); i++)
    // {
    //     cout << temp.at(i).value << " " << temp.at(i).count;
    //     // A*= pow(temp.at(i).value, temp.at(i).count-1)*(temp.at(i).value - 1);
    // }
    ll A = 1;
    for (int i = 0; i < temp.size(); i++)
    {
        // cout << temp.at(i).value << " " << temp.at(i).count;
        A*= pow(temp.at(i).value, temp.at(i).count-1)*(temp.at(i).value - 1);
    }
    cout << A;
}
long long nearest_prime(ll n){

    ll lower = n - 1;
    while (!is_prime(lower) && lower%3 != 0) {
        lower -= 1;
    }
    ll upper = n + 1;
        while (!is_prime(upper) && upper% 3 != 0) {
            upper += 1;
        }
    return abs(n - lower) < abs(n - upper) ? lower : upper;
}
int main()
{
    ll n;
    cout << "input the value of n: "; cin >> n;
```

```
    ETF(n);
    cout << "nearest prime of n and dividable to 3: ";
    cout << nearest_prime(n);
    // cout << "\n the number of coprime number using ETF formula: ";
    // ETFFORMULA(n);
}
```

 # Trả lời: Dán kết quả thực thi vào bên dưới:

```
PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\thi giua ki> g++ bai1.cpp -o bai1
PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\thi giua ki> ./bai1
input the value of n: 403
list:
1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18 19 20 21 22 23 24 25 27 28 29 30 32 33 34 35 36 37 38 40 41 42 43 44 45 46 47 48 49 50 51 53 54 55 56 57 58 59 60 61 63 64 66 67 68 69 7
0 71 72 73 74 75 76 77 79 80 81 82 83 84 85 86 87 88 89 90 92 94 95 96 97 98 99 100 101 102 103 105 106 107 108 109 110 111 112 113 114 115 116 118 119 120 121 122 123 125 126 12
7 128 129 131 132 133 134 135 136 137 138 139 140 141 142 144 145 146 147 148 149 150 151 152 153 154 157 158 159 160 161 162 163 164 165 166 167 168 170 171 172 173 174 175 176
177 178 179 180 181 183 184 185 187 188 189 190 191 192 193 194 196 197 198 199 200 201 202 203 204 205 206 207 209 210 211 212 213 214 215 216 218 219 220 222 223 224 225 226 22
7 228 229 230 231 232 233 235 236 237 238 239 240 241 242 243 244 245 246 249 250 251 252 253 254 255 256 257 258 259 261 262 263 264 265 266 267 268 269 270 271 272 274 275 276
277 278 280 281 282 283 284 285 287 288 289 290 291 292 293 294 295 296 297 298 300 301 302 303 304 305 306 307 308 309 311 313 314 315 316 317 318 319 320 321 322 323 324 326 32
7 328 329 330 331 332 333 334 335 336 337 339 340 342 343 344 345 346 347 348 349 350 352 353 354 355 356 357 358 359 360 361 362 363 365 366 367 368 369 370 371 373 374 375 376
378 379 380 381 382 383 384 385 386 387 388 389 391 392 393 394 395 396 397 398 399 400 401 402
sum:  72540
nearest prime of n and dividable to 3: 402
```

***Câu 2*: (2 điểm)** *Cho hệ phương trình đồng dư sau*

$$\begin{cases} x \equiv 2 \bmod 3 \\ x \equiv 4 \bmod 5 \\ x \equiv 6 \bmod 7 \\ x \equiv 7 \bmod 11 \end{cases}$$

- Viết chương trình C/C++ có sử dụng hàm giải hệ phương trình đồng dư trên.

 # Trả lời: Dán code vào bên dưới:

```
//bai 2
#include <bits/stdc++.h>
using namespace std;
#define MAX 100
#define ll long long
#define fix cout << "a\n"
#define fix1 cout << "a1\n"
#define fix2 cout << "a2\n"
ll Mcalc(ll m[], int size)
{
    ll M = 1;
    for (int i = 0; i<size; i++)
```

```
    {
        M*=m[i];
    }
    return M;
}
ll X_of(ll m[], int index, int size)
{
    ll X = 1;
    int i = 0;
    while (i<size)
    {
        if (i == index)
        {
            i++;


        }
        else{
            X *= m[i];
            i++;
        }
    }
    return X;
}
ll index_invertOf(ll m[], int index, int size)
{
    ll temp = X_of(m,index,size);
    int k = 0;
    // double temp2 = (double)(k*m[index] + 1);
    while ((k*m[index] + 1) % temp != 0)
    {
```

```
        k++;
    }
    ll temp_equal = (k*m[index] + 1) / temp;
    return temp_equal;
}
void ChineseTheorem(int size, ll a[], ll m[])
{
    ll M_temp = Mcalc(m,size);
    // cout << "M = " << M_temp << endl;
    ll S = 0;
    for (int i = 0; i<size; i++)
    {
        ll temp_X = X_of(m, i, size);
        // cout << "tempX " << i << "= " << temp_X << endl;
        ll temp_Y = index_invertOf(m, i, size);
        // cout << "tempY " << i << "= " << temp_Y << endl;
        S += a[i]*temp_X*temp_Y;
    }
    S = S % M_temp;
    while(S % M_temp > M_temp)
    {
        S = S % M_temp;
    }
    cout << " X = " << S << " + " << "k*" << M_temp << endl;
}
int main()
{
    int size;
    cout << "ENTER Number of Equations: ";
    cin >> size; getchar();
```

```
    ll a[size];
    ll m[size];
    for (int i = 0; i<size; i++)
    {
        cout << "a[" << i << "] = "; cin >> a[i];
        cout << "m[" << i << "] = "; cin >> m[i];
    }
    cout << "ALL EQUATIONS: \n";
    for (int i = 0; i<size; i++)
    {
        cout << "x = " << a[i] << " mod " << m[i] << "\n";
    }
    ChineseTheorem(size,a,m);


}
```

 # Trả lời: Dán kết quả thực thi vào bên dưới:

Câu 3 (***3 điểm***): Cho ma trận A. Viết chương trình bằng c/c++ có sử dụng hàm thực hiện phân rã ma trận A.

a) Phân rã LDL^T ma trận A

 # Trả lời: Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 100
void Identify(float a[][MAX] ,int n)
{
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         if (i == j) a[i][j] = 1;
         else a[i][j] = 0;
      }
   }
}
void Reset(float a[][MAX] ,int n)
{
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {

         a[i][j] = 0;
      }
   }
}
void print(float a[][MAX], int n)
{
   cout << "matrix: " << "\n";
   for (int i = 0; i < n; i++)
```

```
    {
        cout << "\n\n";
        for (int j = 0; j < n; j++)
        {
            cout << a[i][j] << "\t\t";
        }
    }
    cout << "\n";
}
void transpose(float a[][MAX], int row, int column, float b[][MAX])
{
    // float b[column][row];
    for (int i = 0; i< row; i++)
    for (int j = 0; j< column; j++)
    {
        b[i][j] = a[j][i];
    }
}
void multiple(float a[][MAX], float b[][MAX], int row_1, int column_1, int row_2, int
column_2, float result[][MAX])
{
    if (column_1 != row_2) return;
    for (int i =0 ; i< row_1; i++)
    {
        for (int j = 0; j < column_2; j++)
        {
            result[i][j] = 0;
            for (int k = 0; k < row_2; k++)
            {
                result[i][j] += a[i][k] * b[k][j];
            }
```

```
    }
  }


}
void LDLT_Decomposition(float matrix[][MAX], float output1[][MAX], float
output2[][MAX], float output3[][MAX], int n)
{
  Identify(output1, n);
  Identify(output3, n);
  // output2[0][0] = matrix[0][0];


  for (int i = 0; i < n; i++) {
    for (int j = 0; j <= i; j++) {
      float sum = 0;


      if (j == i) // summation for diagonals
      {
        for (int k = 0; k < j; k++)
          sum += pow(output1[j][k], 2) * output2[k][k];
        output2[j][j] = matrix[j][j] -
                        sum;
      } else {


        // Evaluating L(i, j) using L(j, j)
        for (int k = 0; k < j; k++)
          sum += (output1[i][k] * output1[j][k] * output2[k][k]);
        output1[i][j] = (matrix[i][j] - sum) /
                        output2[j][j];
      }
    }
  }
```

```
    transpose(output1, n, n, output3);
    cout << setw(15) << " L "
         << setw(31) << "D"
          << setw(32) << "LT" << endl;
    for (int i = 0; i < n; i++) {

        // output1 Triangular
        for (int j = 0; j < n; j++)
            cout << setw(6) << output1[i][j] << "\t";
        cout << "\t";

        // Transpose of output1 Triangular
        for (int j = 0; j < n; j++)
            cout << setw(6) << output2[i][j] << "\t";
        cout << "\t";

        for (int j = 0; j < n; j++)
            cout << setw(6) << output3[i][j] << "\t";
        cout << endl;
    }

}
bool check(float input[][MAX], float L1[][MAX], float L2[][MAX], int n)
{
    float mul[MAX][MAX];

    multiple(L1, L2, n, n, n, n, mul);
    for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
```

```
        if (mul[i][j] != input[i][j]) return false;
    }
    return true;




}
int main()
{
    int n;
    cout << "Enter matrix's size: "; cin>> n;
    float matrix[MAX][MAX];
    for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
        cout << "matrix[" << i << "][" << j << "] = "; cin >> matrix[i][j];
    }
    float L1[MAX][MAX], L2[MAX][MAX], L3[MAX][MAX];


    float Transpose[MAX][MAX];


    cout << "LDLT DECOMPOSITION: \n";
    LDLT_Decomposition(matrix, L1, L2, L3, n);


}
```

Trả lời: Dán kết quả thực thi vào bên dưới với $A = \begin{bmatrix} 4 & 8 & 6 \\ 8 & 8 & 6 \\ 6 & 6 & 6 \end{bmatrix}$ (sai số $\varepsilon = 10^{-5}$):

```
PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\thi giua ki> ./bai3a
Enter matrix's size: 4
matrix[0][0] = 8
matrix[0][1] = 6
matrix[0][2] = 8
matrix[0][3] = 8
matrix[1][0] = 6
matrix[1][1] = 6
matrix[1][2] = 6
matrix[1][3] = 6
matrix[0][0] = 4
matrix[0][1] = 8
matrix[0][2] = 6
matrix[1][0] = 8
matrix[1][1] = 8
matrix[1][2] = 6
matrix[2][0] = 6
matrix[2][1] = 6
matrix[2][2] = 6
LDLT DECOMPOSITION:
            L                       D                       LT
    1       0       0               4       0       0       1       2     1.5
    2       1       0               0      -8       0       0       1    0.75
  1.5    0.75       1               0       0     1.5       0       0       1
PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\thi giua ki>
```

b) Phân rã **eigendecomposition** ma trận A

 # Trả lời: Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 100
void Solve(double a ,double b, double c, double d, double output[])
{


   b /= a;
   c /= a;
   d /= a;


   double disc, q, r, dum1, s, t, term1, r13;
   q = (3.0*c - (b*b))/9.0;
   r = -(27.0*d) + b*(9.0*c - 2.0*(b*b));
   r /= 54.0;
```

```
disc = q*q*q + r*r;
term1 = (b/3.0);


double x1_real, x2_real, x3_real;
double x2_imag, x3_imag;
string x2_imag_s, x3_imag_s;
if (disc > 0)   // One root real, two are complex
{
   s = r + sqrt(disc);
   s = s<0 ? -cbrt(-s) : cbrt(s);
   t = r - sqrt(disc);
   t = t<0 ? -cbrt(-t) : cbrt(t);
   x1_real = -term1 + s + t;
   term1 += (s + t)/2.0;
   x3_real = x2_real = -term1;
   term1 = sqrt(3.0)*(-t + s)/2;
   x2_imag = term1;
   x3_imag = -term1;
   x2_imag_s =  " + "+ to_string(x2_imag) + "i";
   x3_imag_s =  " - "+ to_string(x2_imag) + "i";
}
// The remaining options are all real
else if (disc == 0)  // All roots real, at least two are equal.
{
   x3_imag = x2_imag = 0;
   r13 = r<0 ? -cbrt(-r) : cbrt(r);
   x1_real = -term1 + 2.0*r13;
   x3_real = x2_real = -(r13 + term1);
}
// Only option left is that all roots are real and unequal (to get here, q < 0)
```

```
    else
    {
        x3_imag = x2_imag = 0;
        q = -q;
        dum1 = q*q*q;
        dum1 = acos(r/sqrt(dum1));
        r13 = 2.0*sqrt(q);
        x1_real = -term1 + r13*cos(dum1/3.0);
        x2_real = -term1 + r13*cos((dum1 + 2.0*M_PI)/3.0);
        x3_real = -term1 + r13*cos((dum1 + 4.0*M_PI)/3.0);
    }

    cout << "\nRoots:" << endl <<
            "  x = " << x1_real << endl <<
            "  x = " << x2_real << x2_imag_s << endl <<
            "  x = " << x3_real << x3_imag_s << endl;
    output[0] = x1_real;
    output[1] = x2_real;
    output[2] = x3_real;

}
void sort(double arr[], int n)
{
    int i, j;
    double temp;
        bool swapped;
        for (i = 0; i < n - 1; i++) {
            swapped = false;
            for (j = 0; j < n - i - 1; j++) {
                if (arr[j] < arr[j + 1]) {
```

```
                // Swap arr[j] and arr[j+1]
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = true;
            }
        }

        // If no two elements were
        // swapped by inner loop, then break
        if (swapped == false)
            break;
    }
}
void Init(double a[][MAX], int *n)
{
   fstream File("matrix.txt", ios::in);
   File >> *n;
   for (int i = 0; i < *n; i++)
   for (int j = 0; j < *n; j++)
   {
       File >> a[i][j];
   }

}
void Identify(double a[][MAX] ,int n)
{
   for (int i = 0; i < n; i++)
   {
```

```
        for (int j = 0; j < n; j++)
        {
            if (i == j) a[i][j] = 1.0;
            else a[i][j] = 0.0;
        }
    }
}
void Zerotify(double a[][MAX] ,int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            a[i][j] = 0.0;
        }
    }
}
void Assign(double a[][MAX], double b[][MAX], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            a[i][j] = b[i][j];
        }
    }
}
void print(double a[][MAX], int n)
{
    cout << "matrix: " << "\n";
```

```
    for (int i = 0; i < n; i++)
    {
        cout << "\n\n";
        for (int j = 0; j < n; j++)
        {
            cout  << setprecision(5) << a[i][j] << "\t\t";
        }
    }
    cout << "\n";
}
void multiple(double a[][MAX], double b[][MAX], int n, double result[][MAX])
{

    for (int i =0 ; i< n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            result[i][j] = 0;
            for (int k = 0; k < n; k++)
            {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }

}
void non_squared_multiple(double a[][MAX], double b[][MAX], int row_1, int column_1, int
row_2, int column_2, double result[][MAX])
{
    if (column_1 != row_2) return;
    for (int i =0 ; i< row_1; i++)
```

```
    {
        for (int j = 0; j < column_2; j++)
        {
            result[i][j] = 0;
            for (int k = 0; k < row_2; k++)
            {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }


}
void FormMatrix(double numb, double M[][MAX], double output[][MAX], int row, int
column)
{


    double temp[MAX][MAX];
    Zerotify(temp, row);
    // tinh vector y
    double temp1[MAX][MAX];
    Zerotify(temp1, row);
    for (int i = 0; i<row; i++)
    {
        temp[i][0] = pow(numb, (double)((row-1)-i));
    }
    multiple(M, temp, row, temp1);
    for (int i = 0; i< row; i++)
    {
        output[i][column] = temp1[i][0];
    }
```

```
}

void Eigendecomposite(double a[][MAX], int n)
{
   double M[MAX][MAX], M1[MAX][MAX];
   double temp[MAX][MAX];
   double C[MAX][MAX];
   double P[MAX][MAX];
   Identify(C, n);
   for (int k = n-2; k>=0; k--)
   {
      for (int i = 0; i<n; i++)
      {
         for (int j = 0; j<n; j++)
         {
            if (i != k)
            {
               if (i == j)
               {
                  M[i][j] = 1;
                  M1[i][j] = 1;
               }
               else
               {
                  M[i][j] = 0;
                  M1[i][j] = 0;
               }
            }
            else
```

```
            {
                M1[i][j] = a[k+1][j];
                if (j == k) M[i][j] = 1/a[k+1][k];
                else M[i][j] = -a[k+1][j]/a[k+1][k];
            }
        }
    }

    multiple(a, M, n, temp);
    multiple(M1, temp, n, a);
    multiple(C, M, n, temp);
    Assign(C, temp, n);
  }
  cout << "\n";
  print(a, n);
  Zerotify(temp, n);
  // print(C, n);
  double output[n];
  Solve(1, -a[0][0], -a[0][1], -a[0][2], output);
  sort(output, n); // sap xep gia tri rieng theo thu tu
  for (int i = 0; i<n; i++)
  {

    FormMatrix(output[i], C, temp, n, i);
  }
  print(temp, n);

}

int main(int argc, char const *argv[])
```

```
{
    double matrix[MAX][MAX];
    int n;
    Init(matrix, &n);
    // cout << "Enter Matrix's size: "; cin >> n;
    // for (int i = 0; i < n; i++)
    // for (int j = 0; j < n; j++)
    // {
    //     cout << "matrix[" << i << "][" << j << "] = "; cin >> matrix[i][j];
    // }
    Eigendecomposite(matrix, n);
    return 0;
}
```

Trả lời: Dán kết quả thực thi vào bên dưới với $A = \begin{bmatrix} 4 & 8 & 6 \\ 8 & 8 & 6 \\ 9 & 8 & 6 \end{bmatrix}$ (sai số $\varepsilon = 10^{-5}$):

```
PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\BUOI5> g++ bai1.cpp -o bai3b
PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\BUOI5> ./bai3b

matrix:


18              62              0

1               0               0

0               1               0

Roots:
  x = 20.958
  x = -2.9583
  x = -4.4409e-15
matrix:


0.80738         -1.9272e-15             -1.4489

0.96148         -0.75           0.51022

1               1               1
```

Câu 4 (***3 điểm***): Cho ma trận A. Viết chương trình bằng c/c++ có sử dụng hàm thực hiện phân rã ma trận A bằng phương pháp SVD.

 # Trả lời: Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):

```
//g++ -I C:\eigen-3.4.0 tenfile.cpp -o tenfile.exe
#include <bits/stdc++.h>
#include <Eigen/Dense>
using namespace std;
using namespace Eigen;
MatrixXf Transpose_1(MatrixXf A, int row, int column)
{
    MatrixXf A_trans(column,row);
    for (int i = 0; i< row; i++)
    {
        for (int j = 0; j< column; j++)
```

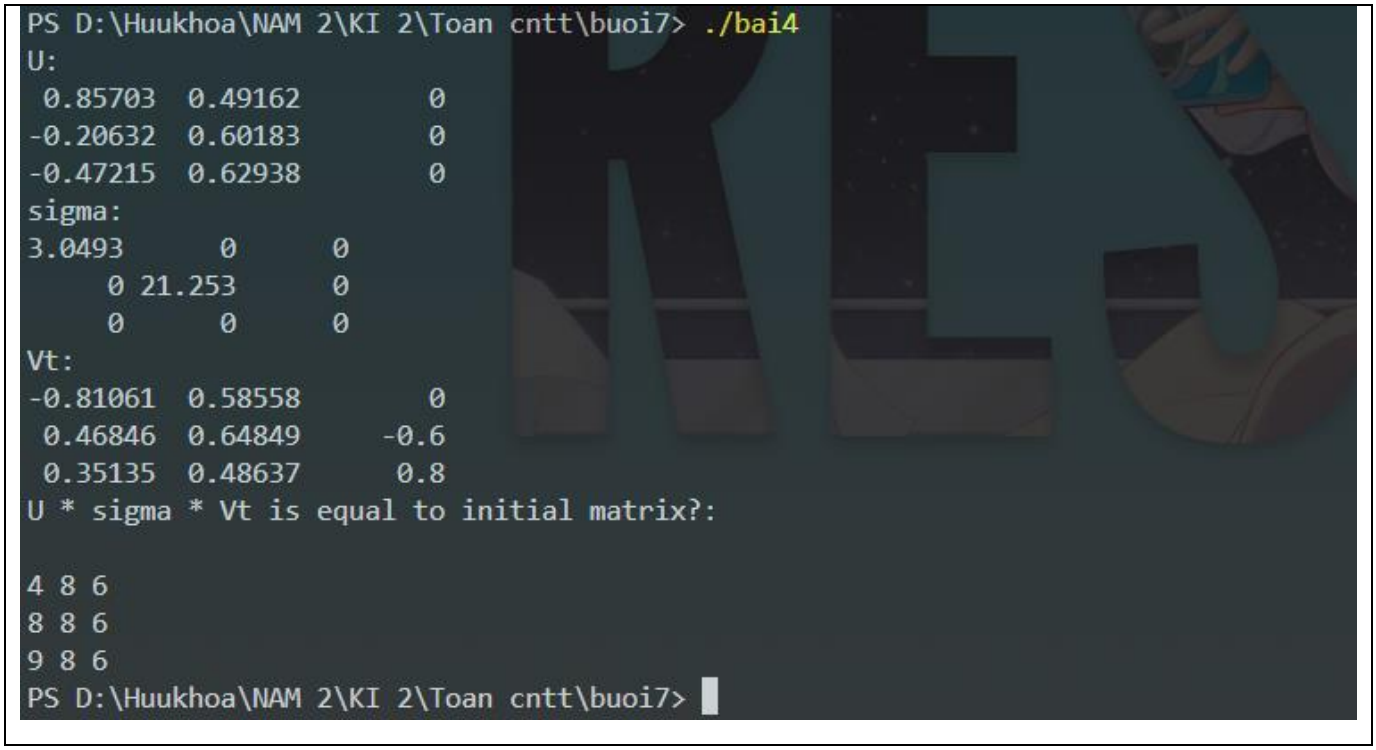
```
        {
            A_trans(j, i) = A(i, j);
        }
    }
    return A_trans;
}
MatrixXf Multiple(MatrixXf A, int A_row, int A_col, MatrixXf B,int B_row, int B_col)
{
    MatrixXf  result(A_col, B_row);
    for (int i =0 ; i< A_col; i++)
    {
      for (int j = 0; j < B_row; j++)
      {
         result(i,j) = 0;
         for (int k = 0; k < B_col; k++)
         {
            result(i,j) += A(i,k)* B(k,j);
         }
      }
    }
   return result;
}
MatrixXf MatrixDacTrung(MatrixXf A, int row, int column)
{
   MatrixXf At = Transpose_1(A, row, column);
   MatrixXf mtdt = Multiple(At, row, column, A, column, row);
   return mtdt;
   // MatrixXf At = A.transpose();
   // MatrixXf mtdt =  At*A;
   // return mtdt;
```

```
}
MatrixXf vtCal(MatrixXf A, int row, int column)
{
   MatrixXf mtdt = MatrixDacTrung(A, row, column);
   EigenSolver<MatrixXf> es(mtdt);
   EigenSolver<MatrixXf>::EigenvectorsType eigenVectors = es.eigenvectors();

   MatrixXf V = eigenVectors.real().matrix();
   return V;
}
MatrixXf sigmaCal(MatrixXf A, int row, int column)
{
   MatrixXf mtdt = MatrixDacTrung(A, row, column);
   EigenSolver<MatrixXf> es(mtdt);
   EigenSolver<MatrixXf>::EigenvalueType eigenValues = es.eigenvalues();
   EigenSolver<MatrixXf>::EigenvectorsType eigenVectors = es.eigenvectors();
   MatrixXf sigma = MatrixXf::Zero(eigenValues.rows(), eigenValues.rows());
    // cout << "\n";
   for (int i = 0; i< eigenValues.rows(); i++)
   {
      sigma(i,i) = sqrt((eigenValues[i].real()));

   }
   // sigma(eigenValues.rows() - 1, eigenValues.rows() - 1) = 0.000001;
    // cout << sigma;
   return sigma;
    //có được ma trận đường chéo lambda
}
MatrixXf uCal(MatrixXf A, int row, int column) //sửa uCal tùy bài
{
```

```
    // MatrixXf mtdt = MatrixDacTrung(A, row, column);
    // EigenSolver<MatrixXf> es(A);
    // EigenSolver<MatrixXf>::EigenvalueType eigenValues = es.eigenvalues();
    // EigenSolver<MatrixXf>::EigenvectorsType eigenVectors = es.eigenvectors();

    MatrixXf V = vtCal(A, row, column);
    // int row1 = V.rows();
    // int col1 = V.cols();

    MatrixXf U = A * V;
    // MatrixXf U = Multiple(A, row, column , V, column, column);
    for (int i = 0; i < U.cols(); ++i)
        U.col(i).normalize();
    return U;
}
void check(MatrixXf U, MatrixXf sigma, MatrixXf Vt)
{
    cout << "\n" <<  (U * sigma * Vt.transpose()).real() << "\n";

}
int main(int argc, char const *argv[])
{
    int row = 3;
    int column = 3;


    MatrixXf A = MatrixXf(row,column);

     A << 4,    8,  6,
          8,    8,  6,
```

```
        9,    8,  6;



    MatrixXf U = uCal(A, row, column);
    cout << "U: " << "\n";
    cout << setprecision(5) <<  U << "\n";
    MatrixXf sigma = sigmaCal(A, row, column);
    cout << "sigma: " << "\n";


    cout << setprecision(5) << sigma << "\n";
    MatrixXf Vt = vtCal(A, row, column);
    cout  <<"Vt: " << "\n";


    cout << setprecision(5) << Vt << "\n";
    cout << "U * sigma * Vt is equal to initial matrix?: \n";
    check(U, sigma, Vt);
    return 0;
}
```

Trả lời: Dán kết quả thực thi vào bên dưới với $A = \begin{bmatrix} 4 & 8 & 6 \\ 8 & 8 & 6 \\ 9 & 8 & 6 \end{bmatrix}$ (sai số $\varepsilon = 10^{-5}$):


PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\buoi7> ./bai4
U:
 0.85703 0.49162   0
-0.20632 0.60183   0
-0.47215 0.62938   0
sigma:
3.0493   0   0
     0 21.253   0
     0   0   0
Vt:
-0.81061 0.58558   0
 0.46846 0.64849   -0.6
 0.35135 0.48637   0.8
U * sigma * Vt is equal to initial matrix?:
4 8 6
8 8 6
9 8 6
PS D:\Huukhoa\NAM 2\KI 2\Toan cntt\buoi7>