LA Assignment

C with OpenMP

Danish Ebadulla , Rahul Raman
PES1201800096 PES1201800146

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

1. i.Gaussian Elimination

Code:

```
#define _POSIX_C_SOURCE 199309L
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
//#include <unistd.h>
#include <omp.h>
#include <stdlib.h>
#include <sys/types.h>
#include <math.h>
#define MAXN 2000  /* Max value of N */
int N;  /* Matrix size */

void parameters(int argc, char **argv) {
 int seed = 0;  /* Random seed */
 char uid[32]; /*User name */
  time_t t;
 /* Read command-line arguments */
 srand((unsigned) time(&t));  /* Randomize */
 if (argc == 3) {
  seed = atoi(argv[2]);
  srand(seed);
  printf("Random seed = %i\n", seed);
 }
 if (argc >= 2) {
  N = atoi(argv[1]);
  if (N < 1 || N > MAXN) {
   printf("N = %i is out of range.\n", N);
   exit(0);
  }
 }
 else {
  printf("Usage: %s <matrix_dimension> [random seed]\n",
       argv[0]);
  exit(0);
 }
 /* Print parameters */
 printf("\nMatrix dimension N = %i.\n", N);
}

double time_elapsed(struct timespec start, struct timespec end) {
  double t;
  t = (end.tv_sec - start.tv_sec);
  t += (end.tv_nsec - start.tv_nsec) * 0.000000001;
  return t;
}

int forward_elimination(int n,float **a){
  for(int i=0;i<n-1;i++){
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
      //#pragma omp parallel for shared(a) private(l,i,k)
      if(a[i][i]==0){
         for(int m=i+1;m<n;m++){
            if(a[m][i]!=0){
               for(int b=0;b<n+1;b++){
                  float temp= a[i][b];
                  a[i][b] = a[m][b];
                  a[m][b] = temp;
               }
               break;
            }
         }
         if(a[i][i]==0) return 0;
      }
      for(int k=1;k<n-i;k++){
         float l = a[i+k][i]/a[i][i];
         for(int j=0;j<n+1;j++){
            a[i+k][j] -= l*a[i][j];
         }
      }
   }
   return 1;
}
int forward_elimination_p(int n,float **a){
   #pragma omp parallel for shared(a) private(l,i,k)
      for(int i=0;i<n-1;i++){
         if(a[i][i]==0){
            for(int m=i+1;m<n;m++){
               if(a[m][i]!=0){
                  for(int b=0;b<n+1;b++){
                     float temp= a[i][b];
                     a[i][b] = a[m][b];
                     a[m][b] = temp;
                  }
                  break;
               }
            }
            if(a[i][i]==0) return 0;
         }
         for(int k=1;k<n-i;k++){
            float l = a[i+k][i]/a[i][i];
            for(int j=0;j<n+1;j++){
               a[i+k][j] -= l*a[i][j];
            }
         }
      }
   return 1;
}

void back_substitution(int n,float **a){
   float sol[n];
   for (int i = n-1; i >= 0; i--)
```

```
  {
    sol[i] = a[i][n];
    for (int j=i+1; j<n; j++)
      sol[i] -= a[i][j]*sol[j];
    sol[i] = sol[i]/a[i][i];
  }
  printf("\nSolution for the system:\n");
  //for (int i=0; i<n; i++)
  //  printf("%f\n", sol[i]);
}

void gauss_elimination(int n,float **a){
  int res = forward_elimination(n,a);
  if(res == 0) printf("\nSingular");
  else{
    int lhs=0;
    int rhs = 1 && a[n-1][n];
    for(int z=0;z<n;z++){
      lhs = lhs || (int)a[n-1][z];
    }
    if(lhs==0 && rhs==0) printf("\nSingular and Infinitely many Solution");
    else if(lhs==0 && rhs!=0) printf("\nSingular and No Solution");
    else {
      back_substitution(n,a);
      printf("\nNon Singular and unique solution");
    }
  }

}

void gauss_elimination_p(int n,float **a){
  int res = forward_elimination_p(n,a);
  if(res == 0) printf("\nSingular");
  else{
    int lhs=0;
    int rhs = 1 && a[n-1][n];
    for(int z=0;z<n;z++){
      lhs = lhs || (int)a[n-1][z];
    }
    if(lhs==0 && rhs==0) printf("\nSingular and Infinitely many Solution");
    else if(lhs==0 && rhs!=0) printf("\nSingular and No Solution");
    else {
      back_substitution(n,a);
      printf("\nNon Singular and unique solution");
    }
  }
}

int main(int argc, char **argv){
  parameters(argc, argv);
  int n=N;
  float **a;
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
    a=malloc(sizeof(float*)*n);
    for(int i=0;i<n;i++){
        a[i]=(float*)malloc(sizeof(float*) * (n+1));
    }
    for (int row = 0; row <n; row++) {
    //initializing A
        for (int col = 0; col <n+1; col++) {
            a[row][col] = (float)rand() / 32768.0;
        }
    }
    struct timespec start, end;
    clock_gettime(CLOCK_REALTIME, &start);
    gauss_elimination(n,a);
    clock_gettime(CLOCK_REALTIME, &end);
    printf("\nTime spent on gaussian_elimination for %d variables sequentially: %lf\n",N,
          time_elapsed(start, end));
    clock_gettime(CLOCK_REALTIME, &start);
    gauss_elimination_p(n,a);
    clock_gettime(CLOCK_REALTIME, &end);
    printf("\nTime spent on gaussian_elimination for %d variables in parallel: %lf\n",N,
          time_elapsed(start, end));
    return 0;
}

}
```

Outputs:

1.Custom 3x3 matrix

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter the number of unknown variables:3
Enter the elements:
1 2 -1 6
2 1 1 3
1 -1 1 -2

Solution for the system:
1.000000
2.000000
-1.000000

Non Singular and unique solution
Time spent on gaussian_elimination for 3 variables sequentially: 0.006982

Singular
Time spent on gaussian_elimination for 3 variables in parallel: 0.004987
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

2.Randomly generated 1000x1000 matrix

```
C:\Users\GF63\Documents\C_programs\Danish>a 1000 3
Random seed = 3

Matrix dimension N = 1000.

Solution for the system:

Non Singular and unique solution
Time spent on gaussian_elimination for 1000 variables sequentially: 1.538849

Solution for the system:

Non Singular and unique solution
Time spent on gaussian_elimination for 1000 variables in parallel: 0.702122
```

1.ii LU Decomposition

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <omp.h>


double time_elapsed(struct timespec start, struct timespec end) {

        double t;

        t = (end.tv_sec - start.tv_sec);

        t += (end.tv_nsec - start.tv_nsec) * 0.000000001;

        return t;

}


int lu(int n,float a[n][n+1]){

  float L[n][n];

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

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

      if(i==j) L[i][j]=1.0f;

      else L[i][j]=0.0f;

    }

  }

  for(int i=0;i<n-1;i++){
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
    if(a[i][i]==0){
       for(int m=i+1;m<n;m++){
          if(a[m][i]!=0){
             for(int b=0;b<n+1;b++){
                float temp= a[i][b];
                a[i][b] = a[m][b];
                a[m][b] = temp;
             }
             break;
          }
       }
       if(a[i][i]==0) return 0;
    }
    for(int k=1;k<n-i;k++){
       float l = a[i+k][i]/a[i][i];
       L[i+k][i] = l;
       for(int j=0;j<n+1;j++){
          a[i+k][j] -= l*a[i][j];
       }
    }
 }
 printf("\n L :=\n");
 for(int i=0;i<n;i++){
    for(int j=0;j<n;j++){
       printf("%f ",L[i][j]);
    }
    printf("\n");
 }
 printf("\n U :=\n");
for(int i=0;i<n;i++){
    for(int j=0;j<n;j++){
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
            printf("%f ",a[i][j]);
        }
        printf("\n");
    }
    return 1;
}

int lu_p(int n,float b[n][n+1]){
    float L[n][n];
    int x=1;
    #pragma omp parallel for
    for (int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(i==j) L[i][j]=1.0f;
            else L[i][j]=0.0f;
        }
    }
    #pragma omp parallel for private(x)
    for(int i=0;i<n-1;i++){
        if(b[i][i]==0){
            for(int m=i+1;m<n;m++){
                if(b[m][i]!=0){
                    for(int x=0;x<n+1;x++){
                        float temp= b[i][x];
                        b[i][x] = b[m][x];
                        b[m][x] = temp;
                    }
                    break;
                }
            }
            if(b[i][i]==0) x=0;
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
        }
        //#pragma omp parallel for
        for(int k=1;k<n-i;k++){
            float l = b[i+k][i]/b[i][i];
            L[i+k][i] = l;
            for(int j=0;j<n+1;j++){
                b[i+k][j] -= l*b[i][j];
            }
        }
    }
    printf("\n L :=\n");
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            printf("%f ",L[i][j]);
        }
        printf("\n");
    }
    printf("\n U :=\n");
   for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            printf("%f ",b[i][j]);
        }
        printf("\n");
    }
    return x;
}



int main(){
    int n;
    printf("Enter the number of unknown variables:");
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
    scanf("%d",&n);
    float a[n][n+1];
    float b[n][n+1];
    printf("Enter the elements:\n");
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            scanf("%f",&a[i][j]);
            b[i][j] = a[i][j];
        }
    }

   struct timespec start, end;
    clock_gettime(CLOCK_REALTIME, &start);
    lu_p(n,a);
    clock_gettime(CLOCK_REALTIME, &end);
    printf("\nTime spent on LU_Decomposition for %d variables in parallel: %lf\n",n,
            time_elapsed(start, end));
    clock_gettime(CLOCK_REALTIME, &start);
    lu(n,b);
    clock_gettime(CLOCK_REALTIME, &end);
    printf("\nTime spent on LU_Decomposition for %d variables in serial: %lf\n",n,
            time_elapsed(start, end));
    return 0;
}
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

Output:

Example 1:

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter the number of unknown variables:4
Enter the elements:
1 3 5 7
9 2 8 11
-1 3 -5 9
2 4 6 -1

 L :=
1.000000 0.000000 0.000000 0.000000
9.000000 1.000000 0.000000 0.000000
-1.000000 -0.240000 1.000000 0.000000
2.000000 0.080000 0.117117 1.000000

 U :=
1.000000 3.000000 5.000000 7.000000
0.000000 -25.000000 -37.000000 -52.000000
0.000000 0.000000 -8.880000 3.520000
0.000000 -0.000000 -0.000000 -11.252253

Time spent on LU_Decomposition for 4 variables in parallel: 0.005984

 L :=
1.000000 0.000000 0.000000 0.000000
9.000000 1.000000 0.000000 0.000000
-1.000000 -0.240000 1.000000 0.000000
2.000000 0.080000 0.117117 1.000000

 U :=
1.000000 3.000000 5.000000 7.000000
0.000000 -25.000000 -37.000000 -52.000000
0.000000 0.000000 -8.880000 3.520000
0.000000 -0.000000 -0.000000 -11.252253

Time spent on LU_Decomposition for 4 variables in serial: 0.006981
```

Example 2:

```
murraman@BLR2-1860008851  /c/Users/murraman/Desktop/LA/github/2.LU_Decomposition
$ python generate_testcase.py
Number of unknown variable:200

murraman@BLR2-1860008851  /c/Users/murraman/Desktop/LA/github/2.LU_Decomposition
$ ./a.exe < input.txt
Enter the number of unknown variables:Enter the elements:

Time spent on LU_Decomposition for 200 variables in parallel: 0.088946

Time spent on LU_Decomposition for 200 variables in serial: 0.123931
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

2. Fundamental Subspaces

Code:

```
#include <stdio.h>

#include <stdlib.h>


void back_substitution(int n,int m,float a[n][m]){
   for (int i=n-1 ; i>=0 ; i--)
   {
      if (a[i][i] != 0)
      {
         for (int k=i-1 ; k>=0 ; k--)
         {
            float l = a[k][i]/a[i][i];
            for(int j=0;j<m;j++){
               //printf("%d %d %d\n",i,j,k);
               a[k][j] -= l*a[i][j];
            }
         }
      }
   }
   printf("\n Echleon Matrix\n");
  for(int i=0;i<n;i++){
      for(int j=0;j<m;j++){
         printf("%0.3f\t",a[i][j]);
      }
      printf("\n");
   }
```

```
}


int forward_elimination(int n,int o,float a[n][o]){
   int c=0;
   for(int i=0;i<n-1;i++){
      if(a[i][i]==0){
         for(int m=i+1;m<n;m++){
            if(a[m][i]!=0){
               for(int b=0;b<o;b++){
                  float temp= a[i][b];
                  a[i][b] = a[m][b];
                  a[m][b] = temp;
               }
               break;
            }
         }
         if(a[i][i]==0) return 0;
      }
      for(int k=1;k<n-i;k++){
         float l = a[i+k][i]/a[i][i];
         for(int j=0;j<o;j++){
            a[i+k][j] -= l*a[i][j];
         }
      }
   }
  for(int i=0;i<n;i++){
     if (i<o && a[i][i]!=0) c++;
   }


   return c;
}
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
void echleon_form(int n,int m,float a[n][m]){
   int res = forward_elimination(n,m,a);
   printf("Column Matrix");
   back_substitution(n,m,a);
   printf("\nRank of Column Space:%d\nBasis of Column space\n",res);
   for (int i=0 ; i<n ; i++)
   {
      for (int j=0 ; j<res ; j++)
      {
         printf("%0.3f\t",a[i][j]);
      }
      printf("\n");
   }
   float trans_a[m][n];
   for (int i=0 ; i<m ; i++){
      for (int j=0 ; j<n ; j++){
         trans_a[i][j] = a[j][i];
      }
   }
   printf("\n\n Transpose of a matrix (Row Matrix)");
   int res1 = forward_elimination(m,n,trans_a);
   back_substitution(m,n,trans_a);
   printf("\nRank of Row Space:%d\nBasis of Row space\n",res1);
   for (int i=0 ; i<n ; i++)
   {
      for (int j=0 ; j<res ; j++)
      {
         printf("%0.3f\t",a[i][j]);
      }
```

```
    printf("\n");
  }
  printf("\nRank of Left Null Space:%d\nBasis of Left Null Space\n",n-res1);
  float left_null_space[n][n-res1];
  if (n-res1 != 0){
  for (int i=0 ; i<n ; i++)
  {
     for (int j=0 ; j<n-res1 ; j++)
     {
        left_null_space[i][j] = 0;
     }
  }
  for (int i=0 ; i<m ; i++)
  {
     if (a[i][i] != 0)
     {
        for (int j = i+1 ; j<n ; j++)
        {
           if (a[i][j] != 0)
           {
              left_null_space[i][n-j-1] = -a[i][j]/a[i][i];
              left_null_space[j][n-j-1] = 1;
           }
        }
     }
  }
  for (int i=0 ; i<n ; i++)
  {
     for (int j=0 ; j<n-res ; j++)
     {
        printf ("%0.3f\t",left_null_space[i][j]);
```

```
    }
    printf("\n");
  }
  }
  else printf("\nNO LEFT NLL SPACE\n");
  printf("\nRank of Null Space:%d\nBasis of Null space\n",m-res);
  float null_space[m][m-res];
  if (m-res !=0){
  for (int i=0 ; i<m ; i++)
  {
    for (int j=0 ; j<m-res ; j++)
    {
      null_space[i][j] = 0;
    }
  }
  for (int i=0 ; i<n ; i++)
  {
    if (a[i][i] != 0)
    {
      for (int j = i+1 ; j<m ; j++)
      {
        if (a[i][j] != 0)
        {
          null_space[i][m-j-1] = -a[i][j]/a[i][i];
          null_space[j][m-j-1] = 1;
        }
      }
    }
  }
  for (int i=0 ; i<m ; i++)
  {
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

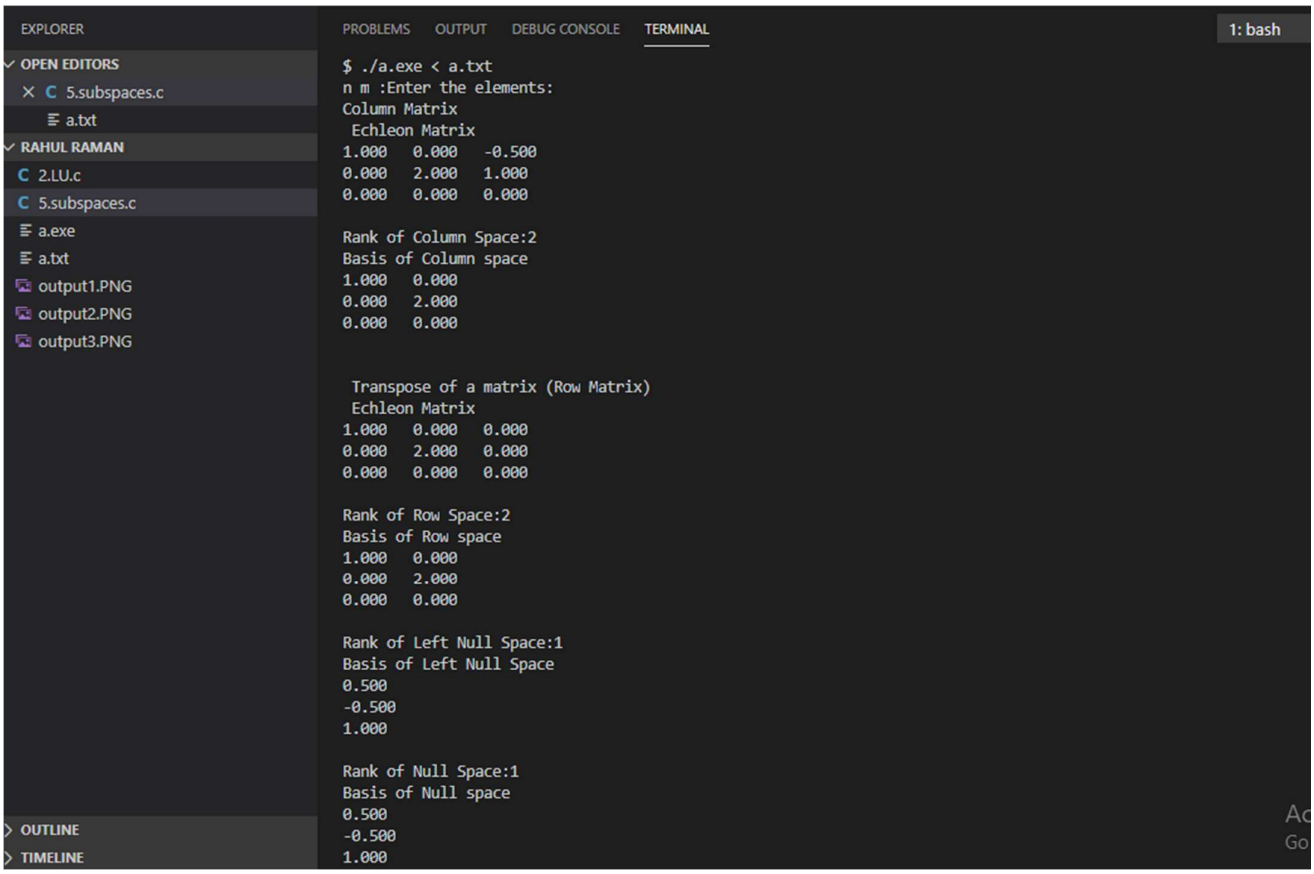
```
      for (int j=0 ; j<m-res ; j++)
      {
         printf ("%0.3f\t",null_space[i][j]);
      }
      printf("\n");
   }
   }
}

int main(){
   int n,m;
   printf("n m :");
   scanf("%d",&n);
   scanf("%d",&m);
   float a[n][m];
   printf("Enter the elements:\n");
   for(int i=0;i<n;i++){
      for(int j=0;j<m;j++){
         scanf("%f",&a[i][j]);
      }
   }
  /*for(int i=0;i<n;i++){
      for(int j=0;j<n+1;j++){
         printf("%f ",a[i][j]);
      }
      printf("\n");
   }
   */
   echleon_form(n,m,a);
   return 0;
}
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

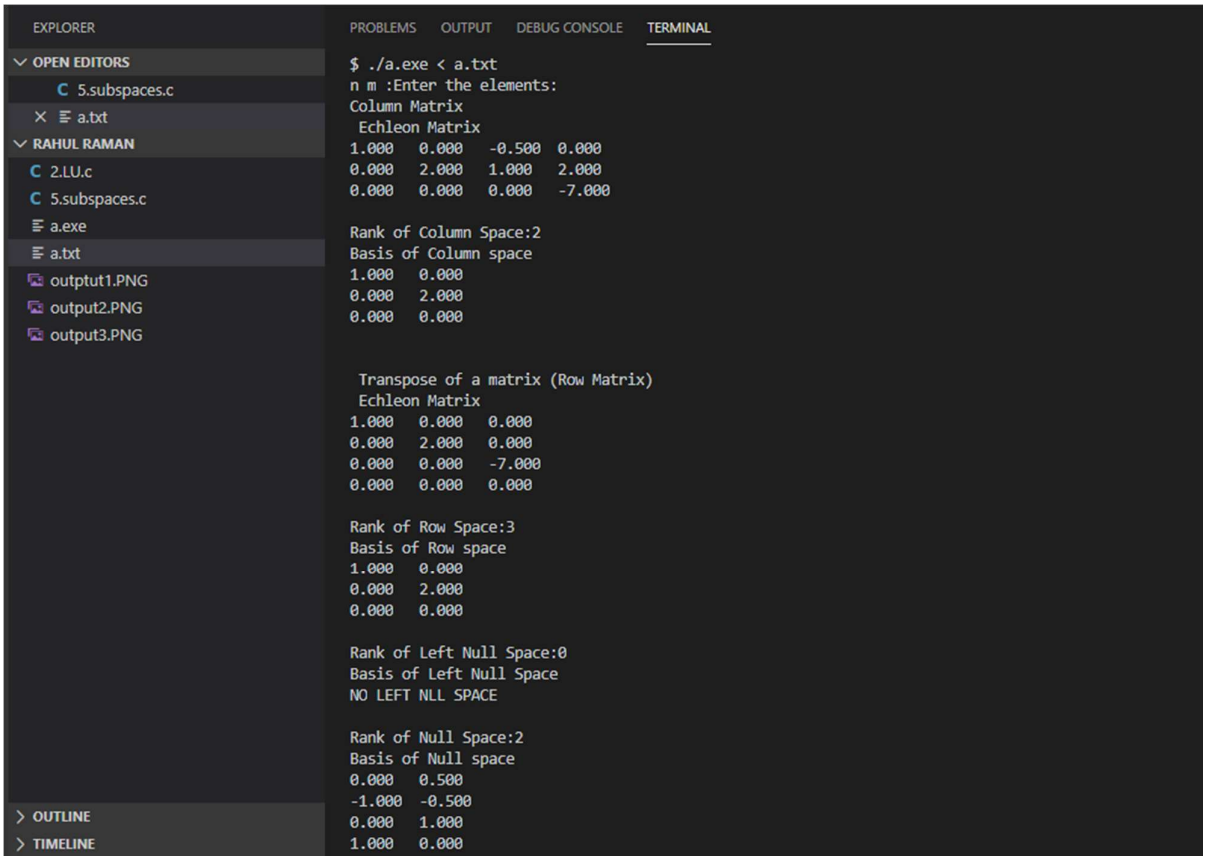
Output:

Example 1:



Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

Example 2:



3.i Computing eigen values and eigen vectors of matrices

Code:

```
#define _POSIX_C_SOURCE 199309L
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
#include <stdlib.h>
#include <sys/types.h>
#include <math.h>
double time_elapsed(struct timespec start, struct timespec end) {
   double t;
   t = (end.tv_sec - start.tv_sec);
   t += (end.tv_nsec - start.tv_nsec) * 0.000000001;
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
    return t;
}
int **makemat(int n,int **a,int c){
    int flag;
    int **b;
    b=malloc(sizeof(int*) *(n-1));
    for(int i=0;i<n-1;i++)
        b[i]=malloc(sizeof(int*) * n-1);
    for(int i=0;i<n-1;i++){
        flag=0;
        for(int j=0;j<n-1;j++){
            if(j==(c-1)){
                b[i][j]=a[i+1][j+1];
                flag=1;
            }
            else
            {
                if(flag)
                    b[i][j]=a[i+1][j+1];
                else
                {
                    b[i][j]=a[i+1][j];
                }
            }
        }
    }
    return b;
}
int determinant(int n,int **a){
    int d=0;
    if(n==1)
        return a[0][0];
    for(int i=1;i<=n;i++){
        int **b=makemat(n,a,i);
        if(i%2==0)
            d-=a[0][i-1] * determinant(n-1,b);
        else if(i%2==1)
            d+=a[0][i-1] * determinant(n-1,b);
    }
    return d;
}
int largest_eigen(int **a,int error,int n){
    float temp, lambda_new, lambda_old;
    float x[n],x_new[n];
    int i,j, step=1;
     for(i=0;i<n;i++)
     {
         if(i==0) x[i]=1;
         else x[i]=0;
     }
     /* Initializing Lambda_Old */
     lambda_old = 1;
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
    /* Multiplication */
    up:
    for(i=0;i<n;i++)
    {
        temp = 0.0;
        for(j=0;j<n;j++)
        {
          temp = temp + a[i][j]*x[j];
        }
        x_new[i] = temp;
    }
    /* Replacing */
    for(i=0;i<n;i++)
    {
      x[i] = x_new[i];
    }
    /* Finding Largest */
    lambda_new = fabs(x[0]);
    for(i=1;i<n;i++)
    {
        if(fabs(x[i])>lambda_new)
        {
          lambda_new = fabs(x[i]);
        }
    }
    /* Normalization */
    for(i=0;i<n;i++)
    {
      x[i] = x[i]/lambda_new;
    }
    if(fabs(lambda_new-lambda_old)>error)
    {
        lambda_old=lambda_new;
        step++;
        goto up;
    }
   for(i=0;i<n;i++)
    {
      printf("%f\t", x[i]);
    }
   return lambda_new;
}
int largest(float *e,int n){
   float x=e[0];
   for(int i=0;i<n;i++){
      if(e[i]>x)
         x=e[i];
   }
   return x;
}
void eigen_values(int **a,int n,int limit){
   int *c[n];
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
   for(int i=0;i<n;i++){
      c[i]= (int*)malloc(n*sizeof(int));
   }
   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
         c[i][j]=a[i][j];
      }
   }
   #pragma omp parallel for
   for(int i= -limit;i!=limit;i++){
      for(int j=0;j<n;j++){
         int temp=a[j][j];
         c[j][j]=temp - i;
      }
      int d=determinant(n,c);
      if(d==0){
         float *e=(float*)malloc(n*sizeof(float));
         for(int i=0;i<n;i++){
            int **f=makemat(n,c,i+1);
            if((i+1)%2==1)
               e[i]=(float)determinant(n-1,f);
            else
               e[i]= -(float)determinant(n-1,f);
         }
         float largest_no=largest(e,n);
         for(int i=0;i<n;i++){
            e[i]=e[i]/largest_no;
         }
         printf("\nEigen vector:\n");
         for(int i=0;i<n;i++){
            printf("%f ",e[i]);
         }
         printf("\n%d is the corresponding eigen value\n",i);
         printf("\n");
      }
   }
}
int main(){
   int n,error=0.001;
   printf("Enter the dimensions of the matrix: ");
   scanf("%d",&n);
   printf("\nEnter the elements:\n");
   int **a;
   a=malloc(sizeof(int*)*n);
   for(int i=0;i<n;i++)
      a[i]=malloc(sizeof(int*) * n);
   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
         scanf("%d",&a[i][j]);
      }
   }
   int D = determinant(n,a);
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
    printf("The determinant is %d\n",D);
    int l_eigen=largest_eigen(a,error,n);
    printf("\n%d is the corresponding eigen value\n",l_eigen);
    struct timespec start, end;
    clock_gettime(CLOCK_REALTIME, &start);
    eigen_values(a,n,l_eigen);
    clock_gettime(CLOCK_REALTIME, &end);
    printf("\nTime spent on eigen value calculation for %dx%d matrix: %lf\n",n,n,
          time_elapsed(start, end));
    return 0;
}
```

Outputs:

3x3 matrix

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter the dimensions of the matrix: 3

Enter the elements:
4 6 10
3 10 13
-2 -6 -8
The determinant is 0
0.875000        1.000000        -0.625000
4 is the corresponding eigen value

Eigen vector:
-1.000000 -1.000000 1.000000
0 is the corresponding eigen value


Eigen vector:
-0.500000 1.000000 -0.500000
2 is the corresponding eigen value


Time spent on eigen value calculation for 3x3 matrix: 0.012965
```

4x4 matrix

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter the dimensions of the matrix: 4

Enter the elements:
2 -1 -1 0
-1 3 -1 -1
-1 -1 3 -1
0 -1 -1 2
The determinant is 0
1.000000        -1.000000       -1.000000       1.000000
4 is the corresponding eigen value

Eigen vector:
1.000000 1.000000 1.000000 1.000000
0 is the corresponding eigen value


Eigen vector:
-1.000000 -0.000000 0.000000 1.000000
2 is the corresponding eigen value


Time spent on eigen value calculation for 4x4 matrix: 0.011968
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

3.ii Computing the largest eigen value using Rayleigh's power method

Code:

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


#define SIZE 10


int main()
{
    float a[SIZE][SIZE], x[SIZE],x_new[SIZE];
    float temp, lambda_new, lambda_old, error;
    int i,j,n, step=1;
    /* Inputs */
    printf("Enter Order of Matrix: ");
    scanf("%d", &n);
    printf("Enter Tolerable Error: ");
    scanf("%f", &error);
    /* Reading Matrix */
    printf("Enter Coefficient of Matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%f", &a[i][j]);
        }
    }
    /* Reading Intial Guess Vector */
    printf("Enter Initial Guess Vector:\n");
    for(i=1;i<=n;i++)
    {
        printf("x[%d]=",i);
        scanf("%f", &x[i]);
    }
    /* Initializing Lambda_Old */
    lambda_old = 1;
    /* Multiplication */
    up:
    for(i=1;i<=n;i++)
    {
        temp = 0.0;
        for(j=1;j<=n;j++)
        {
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
            temp = temp + a[i][j]*x[j];
        }
        x_new[i] = temp;
    }
    /* Replacing */
    for(i=1;i<=n;i++)
    {
      x[i] = x_new[i];
    }
    /* Finding Largest */
    lambda_new = fabs(x[1]);
    for(i=2;i<=n;i++)
    {
        if(fabs(x[i])>lambda_new)
        {
          lambda_new = fabs(x[i]);
        }
    }
    /* Normalization */
    for(i=1;i<=n;i++)
    {
      x[i] = x[i]/lambda_new;
    }
    /* Display */
    printf("\n\nSTEP-%d:\n", step);
    printf("Eigen Value = %f\n", lambda_new);
    printf("Eigen Vector:\n");
    for(i=1;i<=n;i++)
    {
      printf("%f\t", x[i]);
    }
    /* Checking Accuracy */
    if(fabs(lambda_new-lambda_old)>error)
    {
        lambda_old=lambda_new;
        step++;
        goto up;
    }
    return(0);
}
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

Outputs:

3x3 matrix

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter Order of Matrix: 3
Enter Tolerable Error: 0.001
Enter Coefficient of Matrix:
3 -2 0
-2 3 0
0 0 5
Enter Initial Guess Vector:
x[1]=1
x[2]=0
x[3]=0


STEP-1:
Eigen Value = 3.000000
Eigen Vector:
1.000000        -0.666667       0.000000

STEP-2:
Eigen Value = 4.333333
Eigen Vector:
1.000000        -0.923077       0.000000

STEP-3:
Eigen Value = 4.846154
Eigen Vector:
1.000000        -0.984127       0.000000
```

```
STEP-4:
Eigen Value = 4.968254
Eigen Vector:
1.000000        -0.996805       0.000000

STEP-5:
Eigen Value = 4.993610
Eigen Vector:
1.000000        -0.999360       0.000000

STEP-6:
Eigen Value = 4.998720
Eigen Vector:
1.000000        -0.999872       0.000000

STEP-7:
Eigen Value = 4.999744
Eigen Vector:
1.000000        -0.999974       0.000000

STEP-8:
Eigen Value = 4.999949
Eigen Vector:
1.000000        -0.999995       0.000000
C:\Users\GF63\Documents\C_programs\Danish>
```

4x4 matrix

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter Order of Matrix: 4
Enter Tolerable Error: 0.01
Enter Coefficient of Matrix:
3 4 7 12
1 2 -3 9
8 1 2 1
4 -3 2 0
Enter Initial Guess Vector:
x[1]=1
x[2]=1
x[3]=0
x[4]=0


STEP-1:
Eigen Value = 9.000000
Eigen Vector:
0.777778        0.333333        1.000000        0.111111

STEP-2:
Eigen Value = 12.000000
Eigen Vector:
1.000000        -0.046296       0.722222        0.342593

STEP-3:
Eigen Value = 11.981482
Eigen Vector:
1.000000        0.152241        0.812983        0.465997

STEP-4:
Eigen Value = 14.891809
Eigen Vector:
1.000000        0.205449        0.687909        0.347120

STEP-5:
Eigen Value = 12.802594
Eigen Vector:
1.000000        0.193027        0.775498        0.371758

STEP-6:
Eigen Value = 13.661695
Eigen Vector:
1.000000        0.176068        0.740448        0.363931
```

```
STEP-7:
Eigen Value = 13.254582
Eigen Vector:
1.000000        0.181535        0.756033        0.373659

STEP-8:
Eigen Value = 13.502275
Eigen Vector:
1.000000        0.182036        0.745597        0.367898

STEP-9:
Eigen Value = 13.362102
Eigen Vector:
1.000000        0.182483        0.751463        0.370083

STEP-10:
Eigen Value = 13.431172
Eigen Vector:
1.000000        0.181765        0.748668        0.368953

STEP-11:
Eigen Value = 13.395180
Eigen Vector:
1.000000        0.182014        0.750125        0.369688

STEP-12:
Eigen Value = 13.415186
Eigen Vector:
1.000000        0.181947        0.749296        0.369299

STEP-13:
Eigen Value = 13.404444
Eigen Vector:
1.000000        0.182006        0.749739        0.369486

STEP-14:
Eigen Value = 13.410031
Eigen Vector:
1.000000        0.181966        0.749511        0.369385
C:\Users\GF63\Documents\C_programs\Danish>
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

3.iii Solve the normal equation

Code:

```
#define _POSIX_C_SOURCE 199309L
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
#include <stdlib.h>
#include <sys/types.h>
#include <math.h>

float determinant(float **a, float k)
{
 float s = 1, det = 0, **b;
 b=malloc(sizeof(float*)*k);
 for(int i=0;i<k;i++){
  b[i]=(float*)malloc(sizeof(float*) * k);
 }
 int i, j, m, n, c;
 if (k == 1)
  {
   return (a[0][0]);
  }
 else
  {
   det = 0;
   for (c = 0; c < k; c++)
    {
     m = 0;
     n = 0;
     for (i = 0;i < k; i++)
      {
        for (j = 0 ;j < k; j++)
         {
           b[i][j] = 0;
           if (i != 0 && j != c)
            {
              b[m][n] = a[i][j];
              if (n < (k - 2))
               n++;
              else
               {
                n = 0;
                m++;
               }
            }
         }
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
          }
        det = det + s * (a[0][c] * determinant(b, k - 1));
        s = -1 * s;
        }
    }

    return (det);
}

/*Finding transpose of matrix*/
float **transpose(float **num, float **fac, float r)
{
  int i, j;
  float b[25][25],d;
    float **inverse;
    inverse=malloc(sizeof(float*)*r);
    for(int i=0;i<r;i++){
       inverse[i]=(float*)malloc(sizeof(float*) * r);
    }

  for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
        b[i][j] = fac[j][i];
       }
    }
  d = determinant(num, r);
  for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
        inverse[i][j] = b[i][j] / d;
       }
    }
    /*
   printf("\n\n\nThe inverse of matrix is : \n");

   for (i = 0;i < r; i++)
    {
     for (j = 0;j < r; j++)
       {
        printf("\t%f", inverse[i][j]);
       }
    printf("\n");
     }*/
    return inverse;
}

float **cofactor(float **num, int f)
{
 float **b, **fac;
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
 b=malloc(sizeof(float*)*f);
 fac=malloc(sizeof(float*)*f);
 for(int i=0;i<f;i++){
   b[i]=(float*)malloc(sizeof(float*) * f);
   fac[i]=(float*)malloc(sizeof(float*) *f);
 }
 int p, q, m, n, i, j;
 for (q = 0;q < f; q++)
 {
  for (p = 0;p < f; p++)
   {
    m = 0;
    n = 0;
    for (i = 0;i < f; i++)
    {
     for (j = 0;j < f; j++)
      {
       if (i != q && j != p)
       {
        b[m][n] = num[i][j];
        if (n < (f - 2))
         n++;
        else
        {
          n = 0;
          m++;
          }
        }
      }
    }
    fac[q][p] = pow(-1, q + p) * determinant(b, f - 1);
   }
 }
 return transpose(num, fac, f);
}

float **multiplier(int r1, int c1, int r2, int c2,float **first, float **second)
{
   float **mult;
   mult=malloc(sizeof(float*)*r1);
   for(int i=0;i<r1;i++)
      mult[i]=(float*)malloc(sizeof(float*)*c2);
   for (int i = 0; i < r1; ++i) {
      for (int j = 0; j < c2; ++j) {
         mult[i][j] = 0;
      }
   }
   for (int i = 0; i < r1; ++i) {
      for (int j = 0; j < c2; ++j) {
         for (int k = 0; k < c1; ++k) {
            mult[i][j] += first[i][k] * second[k][j];
         }
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
      }
   }
   return mult;
}

int main(){
   float **a;
   float **b;
   int eqns,features;
   printf("Enter the number of equations and features\n");
   scanf("%d %d",&eqns,&features);
   a=malloc(sizeof(float*)*eqns);
   b=malloc(sizeof(float*)*eqns);
   for(int i=0;i<eqns;i++){
      a[i]=(float*)malloc(sizeof(float*) * (features+1));
      b[i]=(float*)malloc(sizeof(float*));
   }
   printf("Enter the values in the format x1 x2... xn y\n");
   for(int i=0;i<eqns;i++){
      for(int j=0;j<=(features+1);j++){
         if(j==0){
            a[i][j]=1;
         }
         else{
            float temp;
            scanf("%f",&temp);
            if(j==(features+1))
               b[i][0]=temp;
            else{
               a[i][j]=temp;
            }
         }
      }
   }
   for(int i=0;i<eqns;i++){
      for(int j=0;j<=features;j++){
         printf("%3.1f ",a[i][j]);
      }
      printf("\n");
   }
   for(int i=0;i<eqns;i++){
      printf("%3.1f ",b[i][0]);
   }
   float **at;
   at=malloc(sizeof(float*)*(features+1));
   for(int i=0;i<eqns;i++){
      at[i]=(float*)malloc(sizeof(float*) * eqns);
   }
   for(int i=0;i<eqns;i++){
      for(int j=0;j<=features;j++){
         at[j][i]=a[i][j];
      }
```

Name - Rahul Raman **SRN - PES1201800146** **Section : 4-A**
Name - Danish Ebadulla **SRN - PES1201800096** **Section : 4-A**
Github_Repo : https://github.com/Aiden-Python/LA---Assignment

```
    }
    printf("\n");
    float **result1=multiplier(features+1,eqns,eqns,features+1,at,a);
    float **result2=multiplier(features+1,eqns,eqns,1,at,b);
    float **result=cofactor(result1,features+1);
    float **result_final=multiplier(features+1,features+1,features+1,1,result,result2);
    printf("The solutions for the least squares line are:\n");
    for(int i=0;i<features;i++){
        printf("%3.1fx%d + ",result_final[i][0],i);
    }
    printf("%3.1fx%d",result_final[features][0],features);
    return 0;
}
```

Output:

Example 1:

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter the number of equations and features
3 1
Enter the values in the format x1 x2... xn y
1 1
2 3
3 2

The solutions for the least squares line are:
1.000x0 + 0.500x1
```

Example 2:

```
C:\Users\GF63\Documents\C_programs\Danish>a
Enter the number of equations and features
5 2
Enter the values in the format x1 x2... xn y
-2 4 0
-1 1 0
0 0 1
1 1 0
2 4 0

The solutions for the least squares line are:
0.486x0 + 0.000x1 + -0.143x2
```