gauss-elimination.c to solve equations

```
#include <stdio.h>

void gauss_elimination(double a[][100], int m,int n, int new2darr[])
{
   int i,j,k,key,flag,track=0;
   double er=0.001,val,x[100],x0[100],sum,a_new[100][100],c;
   for(i=0;i<m;i++)
   {
      x[i] = 0.0;
   }
   for(j=0; j<m; j++)
   {
      for(i=0; i<m; i++)
      {
         if(i>j)
         {
            c=a[i][j]/a[j][j];
            for(k=0; k<m+1; k++)
            {
               a[i][k]=a[i][k]-c*a[j][k];
            }
         }
      }
   }
   x[m-1]=a[m-1][m]/a[m-1][m-1];
   for(i=m-1; i>=0; i--)
   {
      sum=0;
      for(j=i+1; j<m; j++)
      {
         sum=sum+a[i][j]*x[j];
      }
      x[i]=(a[i][m]-sum)/a[i][i];
   }
   for(i=0,j=0;i<n;i++)
   {
      flag = 0;
      for(k=0;k<(n-m);k++)
      {
         if(new2darr[k] == i)
            {
               flag = 1;
            }
      }
      if (flag==0)
      {
         printf(" x%d = %lf ",i+1,x[j++]);
      }
   }
}
```

```
int number_combine(int arr[], int data[], int start, int end,int index, int r, int s)
{
   int i;
   if (index == r)
   {
      return(s+1);
   }

   for (i=start; i<=end && end-i+1 >= r-index; i++)
   {
      data[index] = arr[i];
      s = number_combine(arr, data, i+1, end, index+1, r, s);
   }
   return s;
}


int * combine(int arr[], int data[], int start, int end,int index, int r, int* newarr, int *l)
{
   int j,i;
   if (index == r)
   {
      for (j=0; j<r; j++)
      {
         *(newarr + *l) = data[j];
         (*l)++;
      }
      return newarr;
   }

   for (i=start; i<=end && end-i+1 >= r-index; i++)
   {
      data[index] = arr[i];
      newarr = combine(arr, data, i+1, end, index+1, r, newarr, l);
   }
   return newarr;
}


void main () {

   int i,j,m,n;
   printf("\n Enter number of unknowns(n) : ");
   scanf("%d",&n);
   printf(" Enter number of equations (m) : ");
   scanf("%d",&m);
   double a[m][n],b[m];
   printf("\n");
   for(i=0;i<m;i++)
   {
```

```
    for(j=0;j<n;j++)
    {
        printf(" Input for matrix a's row %d column %d : ",(i+1),(j+1));
        scanf("%lf",&a[i][j]);
    }
}
printf("\n");
for(i=0;i<m;i++)
{
    printf(" Input for matrix B's row %d column 1 : ",(i+1));
    scanf("%lf",&b[i]);
}


int arr[n],k;
for(i=0;i<n;i++)
{
    arr[i] = i;
}
int r = n-m;
int data[r];
int nc = number_combine(arr, data, 0, n-1, 0, r,0);
int *getarr;
int newarr[nc*r+1];
int new2darr[nc+1][r+1];
getarr = newarr;
int l_start = 0;
int *l = &l_start;
getarr = combine(arr, data, 0, n-1, 0, r, getarr, l);
for (i=0;i<nc;i++)
{
    for(j=0;j<r;j++)
    {
        new2darr[i][j] = newarr[i*r+j];
    }
}

double a_new[100][100];
int flag, track, h;

for(i=0;i<nc;i++)
{
    printf("\n");
    for(j=0;j<r;j++)
    {
        printf(" x%d = 0 ",new2darr[i][j]+1);
    }
    track = 0;
    for(j=0;j<n;j++)
    {
        flag = 0;
        for(k=0;k<(n-m);k++)
```

```
        {
            if(new2darr[i][k] == j)
                {
                    flag = 1;
                }
        }
        if (flag==0)
        {
            for(h=0;h<m;h++)
            {
                a_new[h][j-track] = a[h][j];
            }
        }
        else
        {
            track = track+1;
        }
    }
    for(h=0;h<m;h++)
    {
        a_new[h][m] = b[h];
    }
    gauss_elimination(a_new,m,n,new2darr[i]);

  }
}
```

Outputs of gauss-elimination.c

(1)

```
Enter number of unknowns(n) : 4
Enter number of equations (m) : 2

Input for matrix a's row 1 column 1 : 1
Input for matrix a's row 1 column 2 : 1
Input for matrix a's row 1 column 3 : 1
Input for matrix a's row 1 column 4 : 0
Input for matrix a's row 2 column 1 : 2
Input for matrix a's row 2 column 2 : 1
Input for matrix a's row 2 column 3 : 0
Input for matrix a's row 2 column 4 : 1

Input for matrix B's row 1 column 1 : 40
Input for matrix B's row 2 column 1 : 60

x1 = 0  x2 = 0  x3 = 40.000000  x4 = 60.000000
x1 = 0  x3 = 0  x2 = 40.000000  x4 = 20.000000
x1 = 0  x4 = 0  x2 = 60.000000  x3 = -20.000000
x2 = 0  x3 = 0  x1 = 40.000000  x4 = -20.000000
x2 = 0  x4 = 0  x1 = 30.000000  x3 = 10.000000
x3 = 0  x4 = 0  x1 = 20.000000  x2 = 20.000000
```

(2)

```
Enter number of unknowns(n) : 5
Enter number of equations (m) : 3

Input for matrix a's row 1 column 1 : 2
Input for matrix a's row 1 column 2 : 1
Input for matrix a's row 1 column 3 : 1
Input for matrix a's row 1 column 4 : 0
Input for matrix a's row 1 column 5 : 0
Input for matrix a's row 2 column 1 : 1
Input for matrix a's row 2 column 2 : 1
Input for matrix a's row 2 column 3 : 0
Input for matrix a's row 2 column 4 : 1
Input for matrix a's row 2 column 5 : 0
Input for matrix a's row 3 column 1 : 1
Input for matrix a's row 3 column 2 : 0
Input for matrix a's row 3 column 3 : 0
Input for matrix a's row 3 column 4 : 0
Input for matrix a's row 3 column 5 : 1

Input for matrix B's row 1 column 1 : 100
Input for matrix B's row 2 column 1 : 80
Input for matrix B's row 3 column 1 : 40

x1 = 0  x2 = 0  x3 = 100.000000  x4 = 80.000000  x5 = 40.000000
x1 = 0  x3 = 0  x2 = 100.000000  x4 = -20.000000  x5 = 40.000000
x1 = 0  x4 = 0  x2 = 80.000000  x3 = 20.000000  x5 = 40.000000
x1 = 0  x5 = 0  x2 = -nan  x3 = inf  x4 = inf
x2 = 0  x3 = 0  x1 = 50.000000  x4 = 30.000000  x5 = -10.000000
x2 = 0  x4 = 0  x1 = 80.000000  x3 = -60.000000  x5 = -40.000000
x2 = 0  x5 = 0  x1 = 40.000000  x3 = 20.000000  x4 = 40.000000
x3 = 0  x4 = 0  x1 = 20.000000  x2 = 60.000000  x5 = 20.000000
x3 = 0  x5 = 0  x1 = 40.000000  x2 = 20.000000  x4 = 20.000000
x4 = 0  x5 = 0  x1 = 40.000000  x2 = 40.000000  x3 = -20.000000
```

bsf.c to find basic , infeasible & degenerate solutions

```
#include <stdio.h>

void gauss_elimination(double a[][100], int m,int n, int new2darr[])
{
   int i,j,k,key,flag,track=0;
        double er=0.001,val,x[100],x0[100],sum,a_new[100][100],c;
   for(i=0;i<m;i++)
   {
      x[i] = 0.0;
   }
   for(j=0; j<m; j++)
   {
      for(i=0; i<m; i++)
      {
         if(i>j)
         {
            c=a[i][j]/a[j][j];
            for(k=0; k<m+1; k++)
            {
               a[i][k]=a[i][k]-c*a[j][k];
            }
         }
      }
   }
   x[m-1]=a[m-1][m]/a[m-1][m-1];
   for(i=m-1; i>=0; i--)
   {
      sum=0;
      for(j=i+1; j<m; j++)
      {
         sum=sum+a[i][j]*x[j];
      }
      x[i]=(a[i][m]-sum)/a[i][i];
   }
   int check_infeasibility = 0,check_degeneracy = 0;
   for(i=0,j=0;i<n;i++)
   {
      flag = 0;
      for(k=0;k<(n-m);k++)
      {
         if(new2darr[k] == i)
            {
               flag = 1;
            }
      }
      if (flag==0)
      {
         if (x[j] < 0)
         {
            check_infeasibility = 1;
         }
```

```
            printf(" x%d = %lf ",i+1,x[j++]);
        }
    }
    for(i=0;i<(n-m);i++)
    {
        for(j=i+1;j<(n-m);j++)
        {
            if(x[i] == x[j] && x[i] > 0)
            {
                check_degeneracy = 1;
            }
        }
    }
    if(check_infeasibility==1)
    {
        printf(" - Infeasible solution");
    }
    if(check_degeneracy==1)
    {
        printf(" - Degenerate solution");
    }
    if(check_infeasibility==0 && check_degeneracy==0)
    {
        printf(" - Basic feasible solution");
    }
}


int number_combine(int arr[], int data[], int start, int end,int index, int r, int s)
{
    int i;
    if (index == r)
    {
        return(s+1);
    }

    for (i=start; i<=end && end-i+1 >= r-index; i++)
    {
        data[index] = arr[i];
        s = number_combine(arr, data, i+1, end, index+1, r, s);
    }
    return s;
}


int * combine(int arr[], int data[], int start, int end,int index, int r, int* newarr, int *l)
{
            int j,i;
    if (index == r)
    {
        for (j=0; j<r; j++)
                        {
```

```
            *(newarr + *l) = data[j];
            (*l)++;
        }
        return newarr;
    }

    for (i=start; i<=end && end-i+1 >= r-index; i++)
    {
        data[index] = arr[i];
        newarr = combine(arr, data, i+1, end, index+1, r, newarr, l);
    }
    return newarr;
}


void main () {

    int i,j,m,n;
    printf("\n Enter number of unknowns(n) : ");
    scanf("%d",&n);
    printf(" Enter number of equations (m) : ");
    scanf("%d",&m);
    double a[m][n],b[m];
    printf("\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            printf(" Input for matrix a's row %d column %d : ",(i+1),(j+1));
            scanf("%lf",&a[i][j]);
        }
    }
    printf("\n");
    for(i=0;i<m;i++)
    {
        printf(" Input for matrix B's row %d column 1 : ",(i+1));
        scanf("%lf",&b[i]);
    }


    int arr[n],k;
    for(i=0;i<n;i++)
    {
        arr[i] = i;
    }
    int r = n-m;
    int data[r];
    int nc = number_combine(arr, data, 0, n-1, 0, r,0);
    int *getarr;
    int newarr[nc*r+1];
    int new2darr[nc+1][r+1];
    getarr = newarr;
```

```
    int l_start = 0;
    int *l = &l_start;
    getarr = combine(arr, data, 0, n-1, 0, r, getarr, l);
    for (i=0;i<nc;i++)
    {
        for(j=0;j<r;j++)
        {
            new2darr[i][j] = newarr[i*r+j];
        }
    }

    double a_new[100][100];
    int flag, track, h;

    for(i=0;i<nc;i++)
    {
        printf("\n");
        for(j=0;j<r;j++)
        {
            printf(" x%d = 0 ",new2darr[i][j]+1);
        }
        track = 0;
        for(j=0;j<n;j++)
        {
            flag = 0;
            for(k=0;k<(n-m);k++)
            {
                if(new2darr[i][k] == j)
                    {
                        flag = 1;
                    }
            }
            if (flag==0)
            {
                for(h=0;h<m;h++)
                {
                    a_new[h][j-track] = a[h][j];
                }
            }
            else
            {
                track = track+1;
            }
        }
        for(h=0;h<m;h++)
        {
            a_new[h][m] = b[h];
        }
        gauss_elimination(a_new,m,n,new2darr[i]);

    }
}
```

Outputs of bsf.c

(1)

```
Enter number of unknowns(n) : 3
Enter number of equations (m) : 2

Input for matrix a's row 1 column 1 : 2
Input for matrix a's row 1 column 2 : 3
Input for matrix a's row 1 column 3 : 4
Input for matrix a's row 2 column 1 : 3
Input for matrix a's row 2 column 2 : 4
Input for matrix a's row 2 column 3 : 5

Input for matrix B's row 1 column 1 : 5
Input for matrix B's row 2 column 1 : 6


x1 = 0  x1 = -1.000000  x2 = 2.000000  - Infeasible solution
x2 = 0  x0 = -0.500000  x2 = 1.500000  - Infeasible solution
x3 = 0  x0 = -2.000000  x1 = 3.000000  - Infeasible solution
```

(2)

```
Enter number of unknowns(n) : 3
Enter number of equations (m) : 2

Input for matrix a's row 1 column 1 : 2
Input for matrix a's row 1 column 2 : 1
Input for matrix a's row 1 column 3 : 4
Input for matrix a's row 2 column 1 : 3
Input for matrix a's row 2 column 2 : 1
Input for matrix a's row 2 column 3 : 5

Input for matrix B's row 1 column 1 : 11
Input for matrix B's row 2 column 1 : 14

x1 = 0  x2 = -1.000000  x3 = 3.000000  - Infeasible solution
x2 = 0  x1 = 0.500000  x3 = 2.500000  - Basic feasible solution
x3 = 0  x1 = 3.000000  x2 = 5.000000  - Basic feasible solution
```

(3)

```
Enter number of unknowns(n) : 5
Enter number of equations (m) : 2

Input for matrix a's row 1 column 1 : 3
Input for matrix a's row 1 column 2 : 1
Input for matrix a's row 1 column 3 : 5
Input for matrix a's row 1 column 4 : 1
```

Input for matrix a's row 1 column 5 : 0
Input for matrix a's row 2 column 1 : 2
Input for matrix a's row 2 column 2 : 4
Input for matrix a's row 2 column 3 : 1
Input for matrix a's row 2 column 4 : 0
Input for matrix a's row 2 column 5 : 2

Input for matrix B's row 1 column 1 : 12
Input for matrix B's row 2 column 1 : 8

x1 = 0 x2 = 0 x3 = 0 x4 = 12.000000 x5 = 4.000000 - Basic feasible solution
x1 = 0 x2 = 0 x4 = 0 x3 = 2.400000 x5 = 2.800000 - Basic feasible solution
x1 = 0 x2 = 0 x5 = 0 x3 = 8.000000 x4 = -28.000000 - Infeasible solution
x1 = 0 x3 = 0 x4 = 0 x2 = 12.000000 x5 = -20.000000 - Infeasible solution
x1 = 0 x3 = 0 x5 = 0 x2 = 2.000000 x4 = 10.000000 - Basic feasible solution
x1 = 0 x4 = 0 x5 = 0 x2 = 1.473684 x3 = 2.105263 - Basic feasible solution
x2 = 0 x3 = 0 x4 = 0 x1 = 4.000000 x5 = 0.000000 - Basic feasible solution
x2 = 0 x3 = 0 x5 = 0 x1 = 4.000000 x4 = -0.000000 - Basic feasible solution
x2 = 0 x4 = 0 x5 = 0 x1 = 4.000000 x3 = -0.000000 - Basic feasible solution
x3 = 0 x4 = 0 x5 = 0 x1 = 4.000000 x2 = 0.000000 - Basic feasible solution

(4)

Enter number of unknowns(n) : 4
Enter number of equations (m) : 2

Input for matrix a's row 1 column 1 : 2
Input for matrix a's row 1 column 2 : 6
Input for matrix a's row 1 column 3 : 2
Input for matrix a's row 1 column 4 : 1
Input for matrix a's row 2 column 1 : 6
Input for matrix a's row 2 column 2 : 4
Input for matrix a's row 2 column 3 : 4
Input for matrix a's row 2 column 4 : 6

Input for matrix B's row 1 column 1 : 3
Input for matrix B's row 2 column 1 : 2

x1 = 0 x2 = 0 x3 = 2.000000 x4 = -1.000000 - Infeasible solution
x1 = 0 x3 = 0 x2 = 0.500000 x4 = 0.000000 - Basic feasible solution
x1 = 0 x4 = 0 x2 = 0.500000 x3 = 0.000000 - Basic feasible solution
x2 = 0 x3 = 0 x1 = 2.666667 x4 = -2.333333 - Infeasible solution
x2 = 0 x4 = 0 x1 = -2.000000 x3 = 3.500000 - Infeasible solution
x3 = 0 x4 = 0 x1 = 0.000000 x2 = 0.500000 - Basic feasible solution

bsf optimal.c to find out extreme points and optimal objective function value

(1)

```
 Enter number of unknowns (n) : 4
 Enter number of equations (m) : 2

 Input for matrix A's row 1 column 1 : 3
 Input for matrix A's row 1 column 2 : 5
 Input for matrix A's row 1 column 3 : 1
 Input for matrix A's row 1 column 4 : 0
 Input for matrix A's row 2 column 1 : 5
 Input for matrix A's row 2 column 2 : 2
 Input for matrix A's row 2 column 3 : 0
 Input for matrix A's row 2 column 4 : 1

 Input for matrix B's row 1 column 1 : 15
 Input for matrix B's row 2 column 1 : 10

 Input for objective function's coefficient of x1 : 5
 Input for objective function's coefficient of x2 : 3
 Input for objective function's coefficient of x3 : 0
 Input for objective function's coefficient of x4 : 0


 1 : Maximize
 2 : Minimize
 Enter optimization technique for objective function (1 or 2) : 1

 x1 = 0.000000  x2 = 0.000000  x3 = 15.000000  x4 = 10.000000  - Basic feasible solution -
Objective function value = 0.000000
 x1 = 0.000000  x3 = 0.000000  x2 = 3.000000  x4 = 4.000000  - Basic feasible solution - Objective
function value = 9.000000
 x1 = 0.000000  x4 = 0.000000  x2 = 5.000000  x3 = -10.000000  - Infeasible solution
 x2 = 0.000000  x3 = 0.000000  x1 = 5.000000  x4 = -15.000000  - Infeasible solution
 x2 = 0.000000  x4 = 0.000000  x1 = 2.000000  x3 = 9.000000  - Basic feasible solution - Objective
function value = 10.000000
 x3 = 0.000000  x4 = 0.000000  x1 = 1.052632  x2 = 2.368421  - Basic feasible solution - Objective
function value = 12.368421

 Exactly one optimal solution, with optimal objective function value = 12.368421
```

(2)

```
Enter number of unknowns (n) : 6
Enter number of equations (m) : 4

Input for matrix A's row 1 column 1 : 1
Input for matrix A's row 1 column 2 : 2
Input for matrix A's row 1 column 3 : 1
Input for matrix A's row 1 column 4 : 0
Input for matrix A's row 1 column 5 : 0
Input for matrix A's row 1 column 6 : 0
Input for matrix A's row 2 column 1 : 1
Input for matrix A's row 2 column 2 : 1
Input for matrix A's row 2 column 3 : 0
Input for matrix A's row 2 column 4 : 1
Input for matrix A's row 2 column 5 : 0
Input for matrix A's row 2 column 6 : 0
Input for matrix A's row 3 column 1 : 1
Input for matrix A's row 3 column 2 : -1
Input for matrix A's row 3 column 3 : 0
Input for matrix A's row 3 column 4 : 0
Input for matrix A's row 3 column 5 : 1
Input for matrix A's row 3 column 6 : 0
Input for matrix A's row 4 column 1 : 1
Input for matrix A's row 4 column 2 : -2
Input for matrix A's row 4 column 3 : 0
Input for matrix A's row 4 column 4 : 0
Input for matrix A's row 4 column 5 : 0
Input for matrix A's row 4 column 6 : 1

Input for matrix B's row 1 column 1 : 10
Input for matrix B's row 2 column 1 : 6
Input for matrix B's row 3 column 1 : 2
Input for matrix B's row 4 column 1 : 1

Input for objective function's coefficient of x1 : 2
Input for objective function's coefficient of x2 : 1
Input for objective function's coefficient of x3 : 0
Input for objective function's coefficient of x4 : 0
Input for objective function's coefficient of x5 : 0
Input for objective function's coefficient of x6 : 0


1 : Maximize
2 : Minimize
Enter optimization technique for objective function (1 or 2) : 1

x1 = 0.000000  x2 = 0.000000  x3 = 10.000000  x4 = 6.000000  x5 = 2.000000  x6 = 1.000000  -
Basic feasible solution - Objective function value = 0.000000
x1 = 0.000000  x3 = 0.000000  x2 = 5.000000  x4 = 1.000000  x5 = 7.000000  x6 = 11.000000  -
Basic feasible solution - Objective function value = 5.000000
x1 = 0.000000  x4 = 0.000000  x2 = 6.000000  x3 = -2.000000  x5 = 8.000000  x6 = 13.000000  -
```

Infeasible solution
x1 = 0.000000 x5 = 0.000000 x2 = -2.000000 x3 = 14.000000 x4 = 8.000000 x6 = -3.000000 -
Infeasible solution
x1 = 0.000000 x6 = 0.000000 x2 = -0.500000 x3 = 11.000000 x4 = 6.500000 x5 = 1.500000 -
Infeasible solution
x2 = 0.000000 x3 = 0.000000 x1 = 10.000000 x4 = -4.000000 x5 = -8.000000 x6 = -9.000000 -
Infeasible solution
x2 = 0.000000 x4 = 0.000000 x1 = 6.000000 x3 = 4.000000 x5 = -4.000000 x6 = -5.000000 -
Infeasible solution
x2 = 0.000000 x5 = 0.000000 x1 = 2.000000 x3 = 8.000000 x4 = 4.000000 x6 = -1.000000 -
Infeasible solution
x2 = 0.000000 x6 = 0.000000 x1 = 1.000000 x3 = 9.000000 x4 = 5.000000 x5 = 1.000000 -
Basic feasible solution - Objective function value = 2.000000
x3 = 0.000000 x4 = 0.000000 x1 = 2.000000 x2 = 4.000000 x5 = 4.000000 x6 = 7.000000 -
Basic feasible solution - Objective function value = 8.000000
x3 = 0.000000 x5 = 0.000000 x1 = 4.666667 x2 = 2.666667 x4 = -1.333333 x6 = 1.666667 -
Infeasible solution
x3 = 0.000000 x6 = 0.000000 x1 = 5.500000 x2 = 2.250000 x4 = -1.750000 x5 = -1.250000 -
Infeasible solution
x4 = 0.000000 x5 = 0.000000 x1 = 4.000000 x2 = 2.000000 x3 = 2.000000 x6 = 1.000000 -
Basic feasible solution - Objective function value = 10.000000
x4 = 0.000000 x6 = 0.000000 x1 = 4.333333 x2 = 1.666667 x3 = 2.333333 x5 = -0.666667 -
Infeasible solution
x5 = 0.000000 x6 = 0.000000 x1 = 3.000000 x2 = 1.000000 x3 = 5.000000 x4 = 2.000000 -
Basic feasible solution - Objective function value = 7.000000

Exactly one optimal solution, with optimal objective function value = 10.000000