SOURCE CODE

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

int next_iteration_index(double *ptr, int m, int n)
{
   int i,index=0;
   for(i=1;i<n-1;i++)
   {
      if(*(ptr+(m-1)*n+i) < *(ptr+(m-1)*n+index))
      {
         index = i;
      }
   }
   if(*(ptr+(m-1)*n+index) > 0)
   {
      return -1;
   }
   else
   {
      if(*(ptr+(m-1)*n+index) == 0)
      {
         return -2;
      }
   }
   return index;
}

int pivot_index(double *ptr, int m, int n)
{
   int i,j=next_iteration_index(ptr,m,n),index=j;
   double vi = -1.0,vn = -1.0,min = 10000.0;
   for(i=0;i<m-1;i++)
   {
      vi = *(ptr+i*n+j);
      vn = *(ptr+i*n+n-1);
      if(vi>0 && (vn/vi)<min)
      {
         min = vn/vi;
         index = i*n+j;
      }
   }
   return index;
}

double * convert_pivot(double *ptr, int m, int n, int p_index)
{
   double p = *(ptr+p_index);
   p = 1/p;
   *(ptr+p_index) = p;
   return ptr;
```

```
}

double * convert_row(double *ptr, int m, int n, int p_index)
{
   double p = *(ptr+p_index),r;
   int p_row = p_index/n,p_col=p_index%n,i;
   for(i=0;i<n;i++)
   {
      if(i!=p_col)
      {
         r = *(ptr+p_row*n+i);
         r = r*p;
         *(ptr+p_row*n+i) = r;
      }
   }
   return ptr;
}

double * convert_column(double *ptr, int m, int n, int p_index)
{
   double p = *(ptr+p_index),c;
   int p_row = p_index/n,p_col = p_index%n,i;
   for(i=0;i<m;i++)
   {
      if(i!=p_row)
      {
         c = *(ptr+i*n+p_col);
         c = (-1)*c*p;
         *(ptr+i*n+p_col) = c;
      }
   }
   return ptr;
}

double * convert_others(double *ptr, int m, int n, int p_index)
{
   double p = *(ptr+p_index),c,r,s;
   int p_row = p_index/n,p_col = p_index%n,i,j;
   for(i=0;i<m;i++)
   {
      for(j=0;j<n;j++)
      {
         if(i!=p_row && j!=p_col)
         {
            r = *(ptr+p_row*n+j);
            c = *(ptr+i*n+p_col);
            s = *(ptr+i*n+j);
            s += (c*r/p);
            *(ptr+i*n+j) = s;
         }
      }
   }
```

```
    return ptr;
}

void print_array(double *ptr, int m, int n)
{
    int i,j;
    for(i=0;i<m;i++)
    {
        printf("\n");
        for(j=0;j<n;j++)
        {
            printf(" %lf ",*(ptr+i*n+j));
        }
    }
}

void print_solution(double *ptr, int *ptr2, int m, int n)
{
    int i;
    for(i=0;i<m-1;i++)
    {
        if(*(ptr2+i) != -1)
        {
            printf(" x%d = %lf ",*(ptr2+i)+1,*(ptr+i*n+n-1));
        }
    }
    printf("\n Other variables are non-basic, i.e, 0.");
    printf("\n Optimal solution z = %lf ",*(ptr+m*n-1));
    if (next_iteration_index(ptr,m,n)==-2)
    {
        printf("\n Alternate solution exists.");
    }
    else
    {
        printf("\n Unique optimal solution - no alternate solution.");
    }
}

int * swap_variables(int *ptr2, int m, int n, int p)
{
    int p_row = p/n, p_col = p%n;
    if(p_row==m-2)
    {
        *(ptr2+p_col) = 0;
    }
    else
    {
        *(ptr2+p_col) = p_row+1;
    }
    return ptr2;
}
```

```
double * bigm_solve(double *ptr, int *ptr2, int m, int n, int obj_type)
{
   int p,i=0;
   printf("\n\n Tableu from iteration  %d \n",i++);
   print_array(ptr,m,n);
   printf("\n\n");
   while(next_iteration_index(ptr,m,n)!=-1 && i<5)
   {
      printf(" Tableu from iteration  %d \n",i++);
      p = pivot_index(ptr,m,n);
      ptr2 = swap_variables(ptr2,m,n,p);
      ptr = convert_pivot(ptr,m,n,p);
      ptr = convert_row(ptr,m,n,p);
      ptr = convert_column(ptr,m,n,p);
      ptr = convert_others(ptr,m,n,p);
      print_array(ptr,m,n);
      printf("\n\n");
   }
   if(obj_type == 1)
   {
      *(ptr + m*n -1) *= (-1);
   }
   print_solution(ptr,ptr2,m,n);
   return ptr;
}

void main()
{
   int m,n,n_old,i,j,surplus=0,obj_type;
   double M = 1000.0;
   printf("\n Enter number of unknowns (n) : ");
   scanf("%d",&n);
   printf(" Enter number of equations (m) : ");
   scanf("%d",&m);
   m++;
   n++;
   n_old = n;
   int option[m];
   double arr[m*n];
   printf("\n");
   for(i=0;i<m-1;i++)
   {
      for(j=0;j<n-1;j++)
      {
         printf(" Input for marix A's equation %d coeffiecient of x%d : ",(i+1),(j+1));
         scanf("%lf",&arr[n*i+j]);
      }
   }
   printf("\n");
   for(i=0;i<m-1;i++)
   {
      printf(" Input for matrix B's - constant for equation %d : ",(i+1));
```

```
        scanf("%lf",&arr[n*(i+1)-1]);
    }
    printf("\n");
    for(i=0;i<n-1;i++)
    {
        printf(" Input for objective function's coefficient of x%d : ",(i+1));
        scanf("%lf",&arr[(m-1)*n+i]);
    }
    printf("\n");
    for(i=0;i<m-1;i++)
    {
        printf(" Type of equation / inequation %d \n 1. Ax >= B \n 2. Ax = B \n 3. Ax <= B \n Enter
your option : ",(i+1));
        scanf("%d",&option[i]);
        if(option[i]==1)
        {
            n++;
        }
    }

    printf("\n");
    printf(" Type of optimization \n 1. Minimize \n 2. Maximize \n Enter your option : ");
    scanf("%d",&obj_type);

    double bigm_arr[m*n];
    for(i=0;i<m-1;i++)
    {
        for(j=0;j<n_old-1;j++)
        {
            bigm_arr[i*n+j] = arr[i*n_old+j];
        }
    }
    for(i=0;i<m-1;i++)
    {
        if(option[i]==1)
        {
            for(j=n_old-1;j<n-1;j++)
            {
                bigm_arr[i*n+j]= 0.0;
            }
            bigm_arr[i*n+n_old-1+surplus]= -1.0;
            surplus++;
        }
    }
    for(i=0;i<m-1;i++)
    {
        bigm_arr[i*n+n-1] = arr[i*n_old+n_old-1];
    }
    for(j=0;j<n-1;j++)
    {
        bigm_arr[(m-1)*n+j] = arr[(m-1)*n_old+j];
    }
```

```
    double sum;
    bigm_arr[m*n-1] = 0.0;
    for(j=0;j<n;j++)
    {
       bigm_arr[(m-1)*n+j] *= -1;
       sum = 0.0;
       for(i=0;i<m-1;i++)
       {
          if(option[i]<3)
          {
             sum += (bigm_arr[i*n+j]*M);
          }
       }
       bigm_arr[(m-1)*n+j] -= sum;
    }
    double *ptr;
    int arr2[100], *ptr2;
    ptr = bigm_arr;
    ptr2 = arr2;
    ptr = bigm_solve(ptr,ptr2,m,n,obj_type);
}
```

SAMPLE OUTPUT

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

Input for marix A's equation 1 coeffiecient of x1 : 5
Input for marix A's equation 1 coeffiecient of x2 : 7
Input for marix A's equation 1 coeffiecient of x3 : 4
Input for marix A's equation 2 coeffiecient of x1 : 4
Input for marix A's equation 2 coeffiecient of x2 : -7
Input for marix A's equation 2 coeffiecient of x3 : -5
Input for marix A's equation 3 coeffiecient of x1 : 3
Input for marix A's equation 3 coeffiecient of x2 : 4
Input for marix A's equation 3 coeffiecient of x3 : -6

Input for matrix B's - constant for equation 1 : 7
Input for matrix B's - constant for equation 2 : 2
Input for matrix B's - constant for equation 3 : 3

Input for objective function's coefficient of x1 : 3
Input for objective function's coefficient of x2 : 2
Input for objective function's coefficient of x3 : 2

Type of equation / inequation 1
1. Ax >= B
2. Ax = B
3. Ax <= B
Enter your option : 3
Type of equation / inequation 2
1. Ax >= B
```

2. Ax = B
3. Ax <= B
Enter your option : 3
Type of equation / inequation 3
1. Ax >= B
2. Ax = B
3. Ax <= B
Enter your option : 1

Type of optimization
1. Minimize
2. Maximize
Enter your option : 2

Tableu from iteration 0

5.000000 7.000000 4.000000 0.000000 7.000000
4.000000 -7.000000 -5.000000 0.000000 2.000000
3.000000 4.000000 -6.000000 -1.000000 3.000000
-3003.000000 -4002.000000 5998.000000 1000.000000 -3000.000000

Tableu from iteration 1

-0.250000 -1.750000 14.500000 1.750000 1.750000
9.250000 1.750000 -15.500000 -1.750000 7.250000
0.750000 0.250000 -1.500000 -0.250000 0.750000
-1.500000 1000.500000 -5.000000 -0.500000 1.500000

Tableu from iteration 2

-0.017241 -0.120690 0.068966 0.120690 0.120690
8.982759 -0.120690 1.068966 0.120690 9.120690
0.724138 0.068966 0.103448 -0.068966 0.931034
-1.586207 999.896552 0.344828 0.103448 2.103448

Tableu from iteration 3

0.001919 -0.120921 0.071017 0.120921 0.138196
0.111324 -0.013436 0.119002 0.013436 1.015355
-0.080614 0.078695 0.017274 -0.078695 0.195777
0.176583 999.875240 0.533589 0.124760 3.714012

x3 = 0.138196 x1 = 1.015355 x2 = 0.195777
Other variables are non-basic, i.e, 0.
Optimal solution z = 3.714012
Unique optimal solution - no alternate solution.