```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%          ASSIGNMENT 06             %%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% PROBLEM 01 %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


fprintf('\n\n%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n')
fprintf('\nOutput for problem 01:\n')
fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n\n')
disp(' ')


m=4;n=5;A=[-2 0 0 5 3; 3 2 1 2 2;-4 -3 0 -2 6; 5 3 -4 2 6]
Minimax=-2000;Maximin=2000;
for i=1:m
    if(Minimax<min(A(i,:)))
        Minimax=min(A(i,:));
    end
end
for i=1:n
    if(Maximin>max(A(:,i)))
        Maximin=max(A(:,i));
    end
end
if(Minimax==Maximin)
    fprintf('The game value using manual coding is=%d\n\n',Minimax)
else
    fprintf('There is no saddle point\n\n')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Another Way ShortCut %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Minimax=min(max(A))
Maximin=max(min(A'))
if(Minimax==Maximin)
    fprintf('The game value using MATLAB command is= %d\n\n',Minimax)
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% PROBLEM 02(A) %%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\n\n%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n')
fprintf('\nOutput for problem 02(A):\n')
fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n\n')
disp(' ')

clear
m=3;n=4;
a=[2 4 4 1; 10 3 7 7; 6 7 20 5];d=[50 100 150 200];s=[150;200;150];
b=zeros(m,n);
for i=1:m%%%%%%%Finding the matrix:
    for j=1:n
        if(s(i)&& d(j)>0)
            b(i,j)=min(s(i),d(j));
            s(i)=s(i)-b(i,j);d(j)=d(j)-b(i,j);%Subtract
        end
    end
end
fprintf('The cost matrix with northwest rule is,');b
TotalCost=sum(sum(b.*a))



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% PROBLEM 02(B) %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\n\n%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n')
fprintf('\nOutput for problem 02(B):\n')
fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n\n')
disp(' ')

clear
Cost=[2 4 4 1; 10 3 7 7; 6 7 20 5];d=[50 100 150 200];s=[150;200;150];
%Cost=[5 4 3; 8 4 3; 9 7 5];d=[300 200 200];s=[100;300;300];
[m n]=size(Cost);a=Cost;b=zeros(m,n);
while((sum(d)+sum(s)>0))
    %    k=k+1;    fprintf("Round: %d\n\n",k);b %%%%%For Printing the
 Round
    mm=min(min(a));
    [p,q] = find(a==mm,1);
    b(p,q)=min(s(p),d(q));
    s(p)=s(p)-b(p,q);d(q)=d(q)-b(p,q);
    if(s(p)==0)
        a(p,:)=intmax;
    else
        a(:,q)=intmax;
    end
end
fprintf("\n\nUsing LCM:\n\nCost= %d\nCost
 Matrix,",sum(sum(Cost.*b)));b
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% PROBLEM 03 %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


fprintf('\n\n%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n')
fprintf('\nOutput for problem 03:\n')
fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n\n')
disp(' ')

%%%%%%%%%%%%%%%%%%%%  Problem 03  %%%%%%%%%%%%%%%%%%%%

clc
clear
a=[2 4 4 1; 10 3 7 7; 6 7 20 5];d=[50 100 150
 200];s=[150;200;150];m=3;n=4;
Given_Problem=[a s;d sum(d)]
fprintf("The initial Matrix,")  %%%%%%   Initial function created  %%%
[b,Initial_LCM_Cost]=LCM(a,d,s,m,n)%Initial_Solution=[b s;d sum(d)]

fprintf("\n\n\nRoundwise Cij-Ui-Vj Matrix,")
c=optCheck(a,b,m,n) %%%%%%%%%% Optimality   %%%%%%%%%%
while(sum(sum(c<0))~=0)%%%%%%%%%%%%%%   Subtraction Logic without DFS
 %%%%%%%%%%
[p,q]=find(min(min(c))==c,1);
T=0;
for i=1:m
    for j=1:n
        if(b(i,j)~=0 && (i==p||j==q))
            if(i==p)
                for k=1:m
                    if(b(k,q)~=0 && b(k,j)~=0)
                          T=-5;
                          dsub=min(b(k,q),b(i,j));
                          b(p,q)=b(p,q)+dsub;
                          b(k,q)=b(k,q)-dsub;
                          b(k,j)=b(k,j)+dsub;
                          b(i,j)=b(i,j)-dsub;
                          break
                    end
                end
            else
                for k=1:n
                    if(b(k,q)~=0 && b(k,j)~=0)
                          T=-5;
                          dsub=min(b(i,j),b(p,k));
                          b(p,q)=b(p,q)-dsub;
                          b(i,j)=b(i,j)-dsub;
                          b(i,k)=b(i,k)-dsub;
                          b(p,k)=b(p,k)-dsub;
```

```
                        break
                    end
                end

            end
            break
        end
    end
    if(T==-5)
        break
    end
end
fprintf("\n\n\nROundwise Cij-Ui-Vj Matrix,")
c=optCheck(a,b,m,n) %%%%%%%%%% Optimality   %%%%%%%%%%
fprintf("Solution Matrix,")
b
end
fprintf("The optimality condition satisfied and hence the cost is: %d
\n\n",sum(sum(b.*a)))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% PROBLEM 04 %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
fprintf('\n\n%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n')
fprintf('\nOutput for problem 04:\n')
fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n\n')
disp(' ')

clear
a=[20 28 19 13; 15 30 31 28; 40 21 20 17; 21 28 26 12];n=length(a);
fprintf('Row and Column Reduction:')
a=(a'-min(a'))'
a=a-min(a)
[c,ic,sol]=optCheck4(a);
while(ic~=n)
mm=min(min(a+c));
for i=1:n
    for j=1:n
        if(sum(c(i,:)==intmax)+sum(c(:,j)==intmax)==2*n)
            a(i,j)=a(i,j)+mm;
        end
        if(c(i,j)~=intmax)
            a(i,j)=a(i,j)-mm;
        end
    end
```

```
end
[c,ic,sol]=optCheck4(a);
end
Final_Reduced_Matrix=a
Final_Result=sol'

function [c,ic,sol] = optCheck4(c)
ic=0;n=length(c);
for i1=1:2
p=sum(c'==0)';
for i=1:n       %Row Checking
    if(p(i)==1)
        f=find(0==c(i,:));
        c(:,f)=intmax;
        if(f)
            ic=ic+1;sol(i)=f;
        end
    end
end
p=sum(c==0);
for i=1:n       %Column Checking
    if(p(i)==1)
        f=find(0==c(:,i));
        c(f,:)=intmax;
        if(f)
            ic=ic+1;sol(f)=i;
        end
    end
end
end
for i=1:n %Making other elements exactly zero to calculate easily
    for j=1:n
        if(c(i,j)~=intmax)
            c(i,j)=0;
        end
    end
end
end




function[c]=optCheck(a,b,m,n)
c=zeros(m,n);
[p q]=find(0~=b);
u(1:m)=intmax;
v(1:n)=intmax;
u(p(1))=0;%%% Initialisationfor u v
while(sum(intmax==u)+sum(intmax==v)~=0)
    [u v]=solv(u,v,p,q,a); %Solve with a simple function
end
for i=1:m %%%%% Optimality Checking at empty cell
```

```
    for j=1:n
        if(b(i,j)==0)
            c(i,j)=a(i,j)-u(i)-v(j);
        end
    end
end
end

function[u v]=solv(u,v,p,q,a)
k=length(p);
for i=1:k
    if(u(p(i))~=intmax)
        v(q(i))=a(p(i),q(i))-u(p(i));
    end
% end
% for i=1:k
    if(v(q(i))~=intmax)
        u(p(i))=a(p(i),q(i))-v(q(i));
    end
end
end

function [outputArg2,outputArg1] = LCM(Cost,d,s,m,n)
a=Cost;
b=zeros(m,n);
while((sum(d)+sum(s)>0))
    mm=min(min(a));
    [p,q] = find(a==mm,1);
    b(p,q)=min(s(p),d(q));
    s(p)=s(p)-b(p,q);d(q)=d(q)-b(p,q);
    if(s(p)==0)
        a(p,:)=intmax;
    else
        a(:,q)=intmax;
    end
end
sm=sum(sum(Cost.*b));
outputArg1 = sm;
outputArg2 = b;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%
Output for problem 01:
%%%%%%%%%%%%%%%%%%%%%%%%

A =

    -2     0     0     5     3
     3     2     1     2     2
    -4    -3     0    -2     6
     5     3    -4     2     6
```

```
The game value using manual coding is=1


Minimax =

     1


Maximin =

     1

The game value using MATLAB command is= 1



%%%%%%%%%%%%%%%%%%%%%%%%
Output for problem 02(A):
%%%%%%%%%%%%%%%%%%%%%%%%
The cost matrix with northwest rule is,
b =

    50   100     0     0
     0     0   150    50
     0     0     0   150


TotalCost =

        2650



%%%%%%%%%%%%%%%%%%%%%%%%
Output for problem 02(B):
%%%%%%%%%%%%%%%%%%%%%%%%


Using LCM:

Cost= 2700
Cost Matrix,
b =

     0     0     0   150
     0   100   100     0
    50     0    50    50



%%%%%%%%%%%%%%%%%%%%%%%%
Output for problem 03:
%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Given_Problem =

     2     4     4     1   150
    10     3     7     7   200
     6     7    20     5   150
    50   100   150   200   500

The initial Matrix,
b =

     0     0     0   150
     0   100   100     0
    50     0    50    50


Initial_LCM_Cost =

        2700



Roundwise Cij-Ui-Vj Matrix,
c =

     0    -8   -12     0
    17     0     0    15
     0    -9     0     0



ROundwise Cij-Ui-Vj Matrix,
c =

     0     4     0     0
     5     0     0     3
     0     3    12     0

Solution Matrix,
b =

     0     0    50   100
     0   100   100     0
    50     0     0   100

The optimality condition satisfied and hence the cost is: 2100



%%%%%%%%%%%%%%%%%%%%%%%%
Output for problem 04:
%%%%%%%%%%%%%%%%%%%%%%%%
Row and Column Reduction:
```

```
a =

     7    15     6     0
     0    15    16    13
    23     4     3     0
     9    16    14     0


a =

     7    11     3     0
     0    11    13    13
    23     0     0     0
     9    12    11     0


Final_Reduced_Matrix =

     7     8     0     0
     0     8    10    13
    26     0     0     3
     9     9     8     0


Final_Result =

     3
     1
     2
     4

The cost matrix with northwest rule is,
b =

    50   100     0     0
     0     0   150    50
     0     0     0   150


TotalCost =

        2650

   Using LCM:

Cost= 2700
Cost Matrix,
b =

     0     0     0   150
     0   100   100     0
    50     0    50    50
```

Published with MATLAB® R2018b