```
%fibb
format short %Display Output upto 4 Digits
clear all
clc
%Define Objective Function
% f = @(x)(x<1/2).*((1-x)/2)+(x>=1/2).*(x.^2);
%make fn -ve for maxi
f=@(x)(x*(x-2));
% L=-1;%Lower Limit
% R=1; %Upper Limit
L=0;
R=1.5;
n=6; %Number of Iterations
%Compute Fibonacci Series
Fib = ones(1,2);
for i=3:n+1
    Fib(i)=Fib(i-1)+Fib(i-2);
end
%Task is to construct the table
for k=1:n
    ratio = Fib(n+1-k)/Fib(n+1-k+1);
    x2  = L+ratio.*(R-L); %Computing x2
    x1  = L+R-x2;         %Computing x1
    fx1 = f(x1);
    fx2 = f(x2);
    %Printing the table
    rsl(k,:)=[L R x1 x2 fx1 fx2];
    %Finf minimum between them
    if fx1<=fx2
        R=x2;
    else
        L=x1;
    end
end
%Printing the Results
Variables={'L','R','x1','x2','f(x1)','f(x2)'};
Resl=array2table(rsl);
Resl.Properties.VariableNames(1:size(Resl,2))=Variables

optimalValueX = (L+R)/2;
fprintf('\nOptimal value of x is %f',optimalValueX);
fprintf('\nOptimal value of f(x) is %f \n',f(optimalValueX));
```

```
%lcm
format short
clear all
clc
```

```
Cost = [2 10 4 5;6 12 8 11; 3 9 5 7];
A = [12 25 20]; %Row
B = [25 10 15 5]; %%Column

%%Checked Balanced/Unbalanced Problem
if sum(A) == sum(B)
    fprintf('Given Transportation Problem is Balanced\n');
else
    fprintf('Given Transportation Problem is Unbalanced\n');
    %%Add Dummy row or column
    if sum(A)<sum(B) %Adding Dummy Row
        Cost(end+1,:) = zeros(1,size(A,2));
        A(end+1)=sum(B)-sum(A);
    elseif sum(B)<sum(A) %adding Dummy Column
        Cost(:,end+1) = zeros(size(A,1),1);
        B(end+1)=sum(A)-sum(B);
    end
end

ICost = Cost; %Save the Cost Copy
X = zeros(size(Cost)); %Initialize Allocation
[m,n] = size(Cost); %Finding Rows-Column
BFS = m+n-1; %Total BFS
for i=1:size(Cost,1)
    for j=1:size(Cost,2)
        hh=min(Cost(:));%Finding min Cost Value
        [rowind, colind] = find(hh==Cost);
        x11 = min(A(rowind), B(colind));
        [val, ind] = max(x11); %Find Max Allocation
        ii = rowind(ind); %Identify Row Position
        jj = colind(ind); %Identify Col Position
        y11 = min(A(ii), B(jj)); %Find the Value
        %We set the allocation at (ii, jj) position
        X(ii,jj) = y11;
        A(ii) = A(ii)-y11; %Reduce Row Value
        B(jj) = B(jj)-y11; %Reduce Col Value
        %Assigning the current min value as very high value
        Cost(ii, jj) = Inf; %Cell Covered
    end
end

%Print Initial BFS
fprintf('Initial BFS = \n');
IB = array2table(X);
disp(IB);
%Check for degenerate or Non Degenerate
TotalBFS = length(nonzeros(X));
if TotalBFS==BFS
    fprintf('Initial BFS is Non-Degenerate \n');
else
    fprintf('Initial BFS is Degenerate \n');
end
%Printing Total Transportation Cost
TotalCost = 0;
```

```
for i=1:size(Cost,1)
    for j=1:size(Cost,2)
        TotalCost = TotalCost + X(i,j) * ICost(i,j);
    end
end
fprintf('Total Transportation Cost will be %d\n',TotalCost);




%dual simplex
format short;
clear all
clc
%less than for max

Variable={'x1','x2','s1','s2','sol'};
Cost = [-3 -5 0 0 0];
Info = [-1 -3; -1 -1];
b = [-3; -2];

s=eye(size(Info,1));

A = [Info s b];

%Finding starting BFS
BV=[];
for j=1:size(s,2)
    for i=1:size(A,2)
        if A(:,i)==s(:,j)
            BV = [BV i];
        end
    end
end

fprintf('Basic Variables (BV) =');
disp(Variable(BV));

zjcj=Cost(BV)*A - Cost;

%for Print Table
ZCj = [zjcj;A];
SimpTable=array2table(ZCj);
SimpTable.Properties.VariableNames(1:size(ZCj,2)) = Variable
%%Dual-Simplex method start
RUN = true;
while RUN
    SOL = A(:, end);
    if any (SOL<0)
        fprintf('The Current Solution is NOT FEASIBLE \n');
        [LeaVal, pvt_row] = min(SOL);
        fprintf('Leaving Row = %d \n',pvt_row);
        %Finding the entering variable
        ROW = A(pvt_row, 1:end-1);
        ZJ = zjcj(:,1:end-1);
        for i=1:size(ROW,2)
            if ROW(i)<0
                ratio(i) = abs(ZJ(i)./ROW(i));
            else
                ratio(i) = inf;
            end
        end
        %Finding The Entering Variable
        [minVAL, pvt_col]=min(ratio);
        fprintf('Entering Variable = %d ',pvt_col);
        disp(Variable(pvt_col));
        %Update the BV
        BV(pvt_row) = pvt_col;
        fprintf('Basic Variables (BV) =');
        disp(Variable(BV));
        %Update the Table for next iteration
        pvt_key = A(pvt_row,pvt_col);
        A(pvt_row,:) = A(pvt_row,:)./pvt_key;
        for i=1:size(A,1)
            if i~=pvt_row
                A(i,:)=A(i,:)-A(i,pvt_col).*A(pvt_row,:);
            end
        end

        zjcj=Cost(BV)*A - Cost;

        %for Print Table
        ZCj = [zjcj;A];
        SimpTable=array2table(ZCj);
        SimpTable.Properties.VariableNames(1:size(ZCj,2)) = Variable
    else
        RUN=false;
        fprintf('CURRENT BFS IS Feasible & Optimal \n');
    end
end
%FINAL OPTIMAL SOLUTION PRINT
FINAL_BFS = zeros(1,size(A,2));
FINAL_BFS(BV) = A(:,end);
FINAL_BFS(end) = sum(FINAL_BFS.*Cost);
%change - again if
OptimalBFS = array2table(FINAL_BFS);
OptimalBFS.Properties.VariableNames(1:size(OptimalBFS,2))=Variable


%multiobjective
format short
clear all
clc
%Define Variable
%for max +convert to less than,-z for min then at end again -
Z_all=[1 4 1; 2 7 5;];
A = [1 1 1; -1 -5 -4;];
B = [8; -15];
%Converting multi object problem in linear problem
z=[];
```

```
for j=1:size(Z_all,2)
    row = size(Z_all,1);
    sum=0;
    for i=1:row
        sum=sum+Z_all(i,j);
    end
    z(j)=sum/row;
end
%Solving with the simplex method
m=size(A,1);
n=size(A,2);
s=eye(m);
A=[A s B];
m=size(A,1);
n=size(A,2);
Cost = zeros(1,n);
Cost(1:size(z,2)) = z;
%Basic Variables
BV = size(z,2)+1:size(A,2)-1;
%Calculate zjcj
zjcj = Cost(BV)*A - Cost;

%Printing the table
zjcjA = [zjcj;A];
Variables = {'x1','x2','x3','s1','s2','Sol'};
zjA = array2table(zjcjA);
zjA.Properties.VariableNames = Variables

%
RUN = true;
while RUN
    if any(zjcj<0)
        disp('Not an Optimal Solution');
        [pivotVal, pivotCol]=min(zjcj(1:size(zjcj,2)-1));
        sol = A(:,end);
        Column = A(:,pivotCol);
        for i=1:m
            if Column(i)>0
                ratio(i) = sol(i)/Column(i);
            else
                ratio(i) = inf;
            end
        end
        [minRatio, pivotRow] = min(ratio);
        pivotKey=A(pivotRow,pivotCol);
        BV(pivotRow)=pivotCol;
        A(pivotRow,:)=A(pivotRow,:)./pivotKey;
        for i=1:size(A,1)
            if i~=pivotRow
                A(i,:)=A(i,:)-A(i,pivotCol).*A(pivotRow,:);
            end
        end
        %Calculate zjcj
        zjcj = Cost(BV)*A - Cost;
        %Printing the table
        zjcjA = [zjcj;A];
        Variables = {'x1','x2','x3','s1','s2','Sol'};
        zjA = array2table(zjcjA);
        zjA.Properties.VariableNames = Variables
    else
        RUN = false;
        disp('Optimal Solution Reached');
        fprintf('BASIC VARIABLES ARE :');
        disp(Variables(BV));
        fprintf('Maximize Z = %d \n', zjcj(end));
        BFS(BV) = A(:,end);
        BFS(end+1) = zjcj(end);
        cuuBFS = array2table(BFS);
        cuuBFS.Properties.VariableNames = Variables
    end
end
```