GD GOENKA SCHOOL OF ENGINEERING

GD GOENKA UNIVERSITY
FROM FOUNDATION TO FLIGHT

Numerical Methods Practical File
(MAT2002L)

School of Engineering

Department of Computer Science Engineering

Submitted to:

Dr. Naresh Sharma

Dr. Sandeep Nagar

School of Engineering

Submitted by:

Shubham Chahal

(140020203052)

B.Tech. CSE, 4th Semester

INDEX

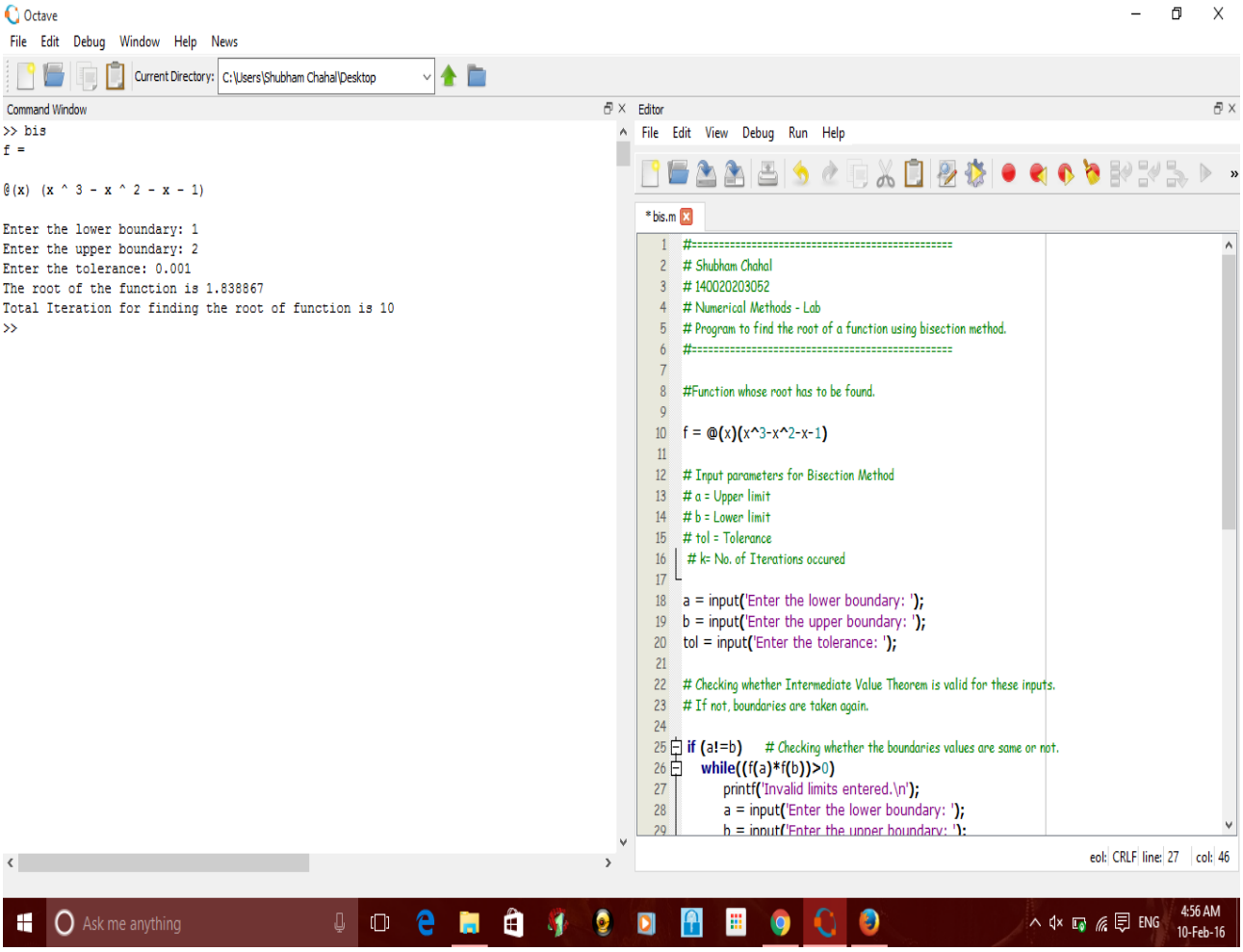
Write a Program to implement Bisection Method.

```
#=================================================
# Shubham Chahal
# 140020203052
# Numerical Methods - Lab
# Program to find the root of a function using bisection method.
#=================================================
#Function whose root has to be found.
f = @(x)(x^3-x^2-x-1)

# Input parameters for Bisection Method
# a = Upper limit
# b = Lower limit
# tol = Tolerance
 # k= No. of Iterations occured

a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');

# Checking whether Intermediate Value Theorem is valid for these inputs.
# If not, boundaries are taken again.


 if (a!=b)     # Checking whether the boundaries values are same or not.
    while((f(a)*f(b))>0)
        printf('Invalid limits entered.\n');
        a = input('Enter the lower boundary: ');
        b = input('Enter the upper boundary: ');
        tol = input('Enter the tolerance: ');
```

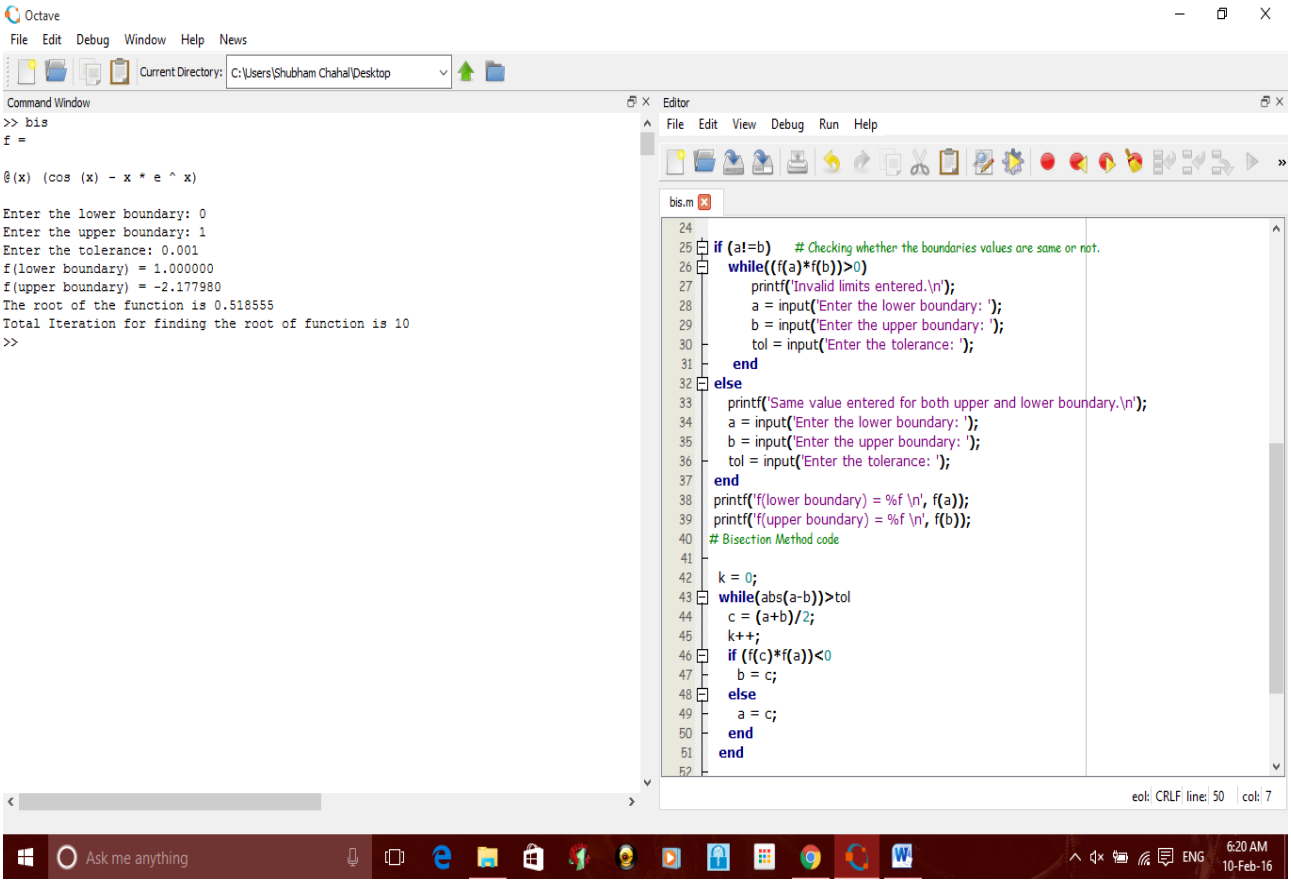
```
    end
else
    printf('Same value entered for both upper and lower boundary.\n');
    a = input('Enter the lower boundary: ');
    b = input('Enter the upper boundary: ');
    tol = input('Enter the tolerance: ');
end

# Bisection Method code
 k = 0;
 while(abs(a-b))>tol
    c = (a+b)/2;
    k++;
    if (f(c)*f(a))<0
      b = c;
    else
      a = c;
    end
 end

 # Display the value of c, i.e root of the function.
printf('The root of the function is %f \n',c);

# Display the value of k,, i.e the number of iterations occured.
printf('Total Iteration for finding the root of function is %d\n',k);
```

Outputs:

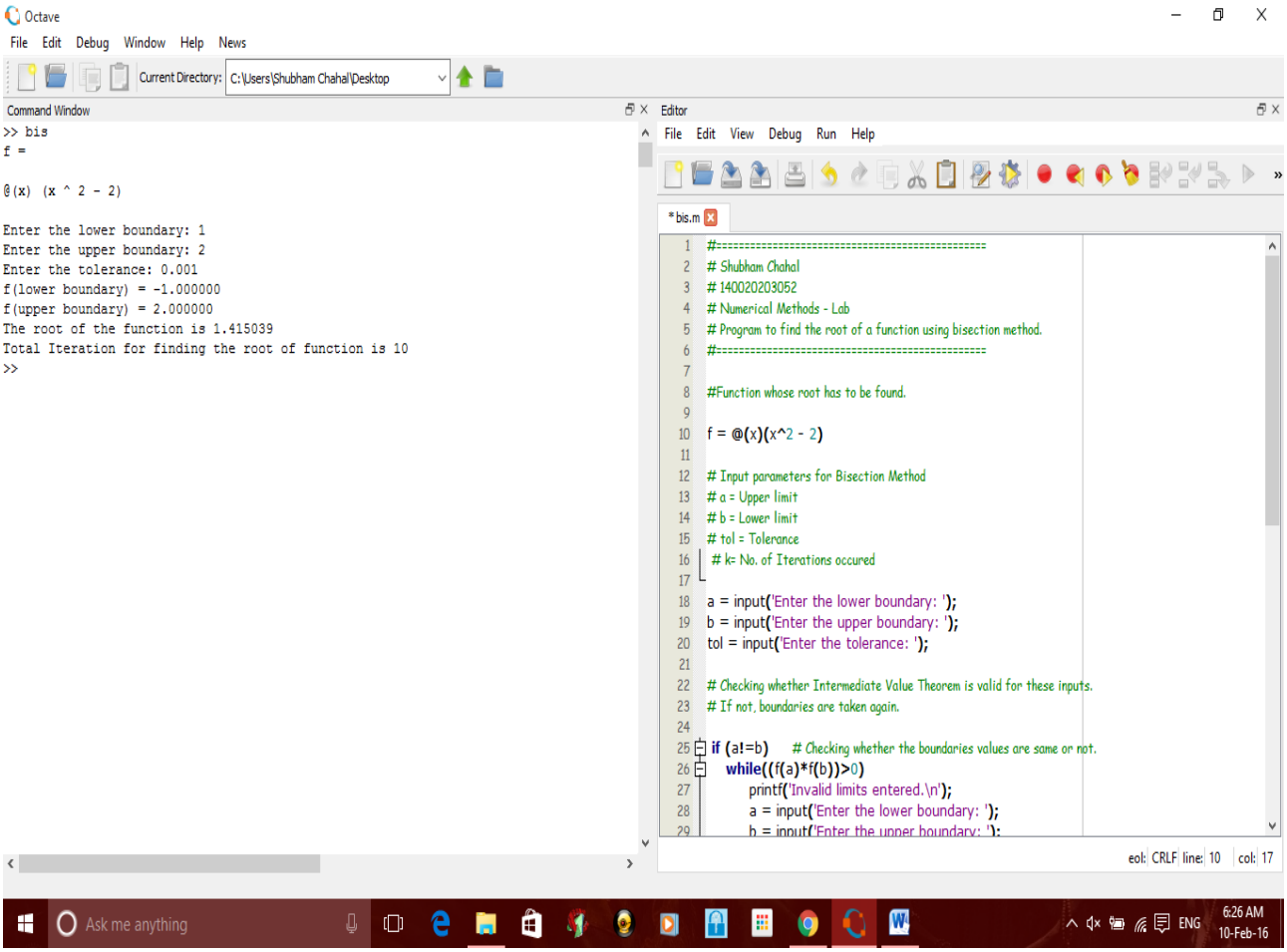
a). f(x) = $x^3 - x^2 - x - 1$



b). $f(x) = \cos(x) - x.e^x$


Octave
File Edit Debug Window Help News
Current Directory: C:\Users\Shubham Chahal\Desktop
Command Window
>> bis
f =
@(x) (cos (x) - x * e ^ x)
Enter the lower boundary: 0
Enter the upper boundary: 1
Enter the tolerance: 0.001
f(lower boundary) = 1.000000
f(upper boundary) = -2.177980
The root of the function is 0.518555
Total Iteration for finding the root of function is 10
>>
Editor
File Edit View Debug Run Help
bis.m
if (a!=b) # Checking whether the boundaries values are same or not.
while((f(a)*f(b))>0)
printf('Invalid limits entered.\n');
a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');
end
else
printf('Same value entered for both upper and lower boundary.\n');
a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');
end
printf('f(lower boundary) = %f \n', f(a));
printf('f(upper boundary) = %f \n', f(b));
# Bisection Method code
k = 0;
while(abs(a-b))>tol
c = (a+b)/2;
k++;
if (f(c)*f(a))<0
b = c;
else
a = c;
end
end
eol: CRLF line: 50 col: 7
Ask me anything
ENG
6:20 AM
10-Feb-16

c). $f(x) = x^2 - 2$


Octave
File Edit Debug Window Help News
Current Directory: C:\Users\Shubham Chahal\Desktop
Command Window
>> bis
f =
@(x) (x ^ 2 - 2)
Enter the lower boundary: 1
Enter the upper boundary: 2
Enter the tolerance: 0.001
f(lower boundary) = -1.000000
f(upper boundary) = 2.000000
The root of the function is 1.415039
Total Iteration for finding the root of function is 10
>>
Editor
File Edit View Debug Run Help
*bis.m
#==========
Shubham Chahal
140020203052
Numerical Methods - Lab
Program to find the root of a function using bisection method.
#==========
#Function whose root has to be found.
f = @(x)(x^2 - 2)
# Input parameters for Bisection Method
# a = Upper limit
# b = Lower limit
# tol = Tolerance
# k= No. of Iterations occured
a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');
# Checking whether Intermediate Value Theorem is valid for these inputs.
# If not, boundaries are taken again.
if (a!=b) # Checking whether the boundaries values are same or not.
while((f(a)*f(b))>0)
printf('Invalid limits entered.\n');
a = input('Enter the lower boundary: ');
eol: CRLF line: 10 col: 17
Ask me anything
ENG
6:26 AM
10-Feb-16

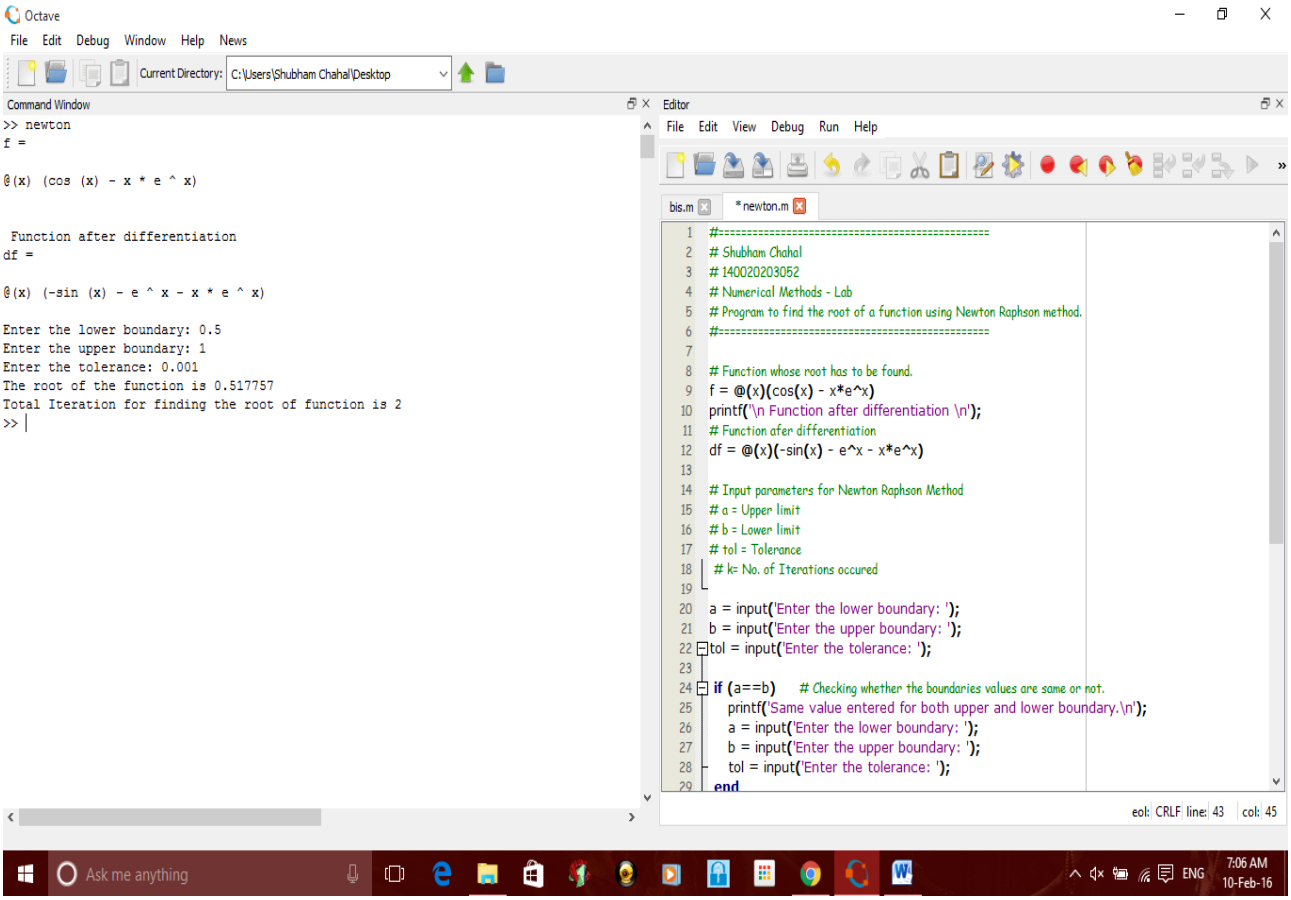
Write a Program to implement Newton Raphson Method.

```
#================================================
# Shubham Chahal
# 140020203052
# Numerical Methods - Lab
# Program to find the root of a function using Newton Raphson method.
#================================================

# Function whose root has to be found.
f = @(x)(cos(x) - x*e^x)
printf('\n Function after differentiation \n');
# Function after differentiation
df = @(x)(-sin(x) - e^x - x*e^x)

# Input parameters for Newton Raphson Method
# a = Upper limit
# b = Lower limit
# tol = Tolerance
 # k= No. of Iterations occurred

a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');

 if (a==b)     # Checking whether the boundaries values are same or not.
    printf('Same value entered for both upper and lower boundary.\n');
    a = input('Enter the lower boundary: ');
    b = input('Enter the upper boundary: ');
    tol = input('Enter the tolerance: ');
 end
```

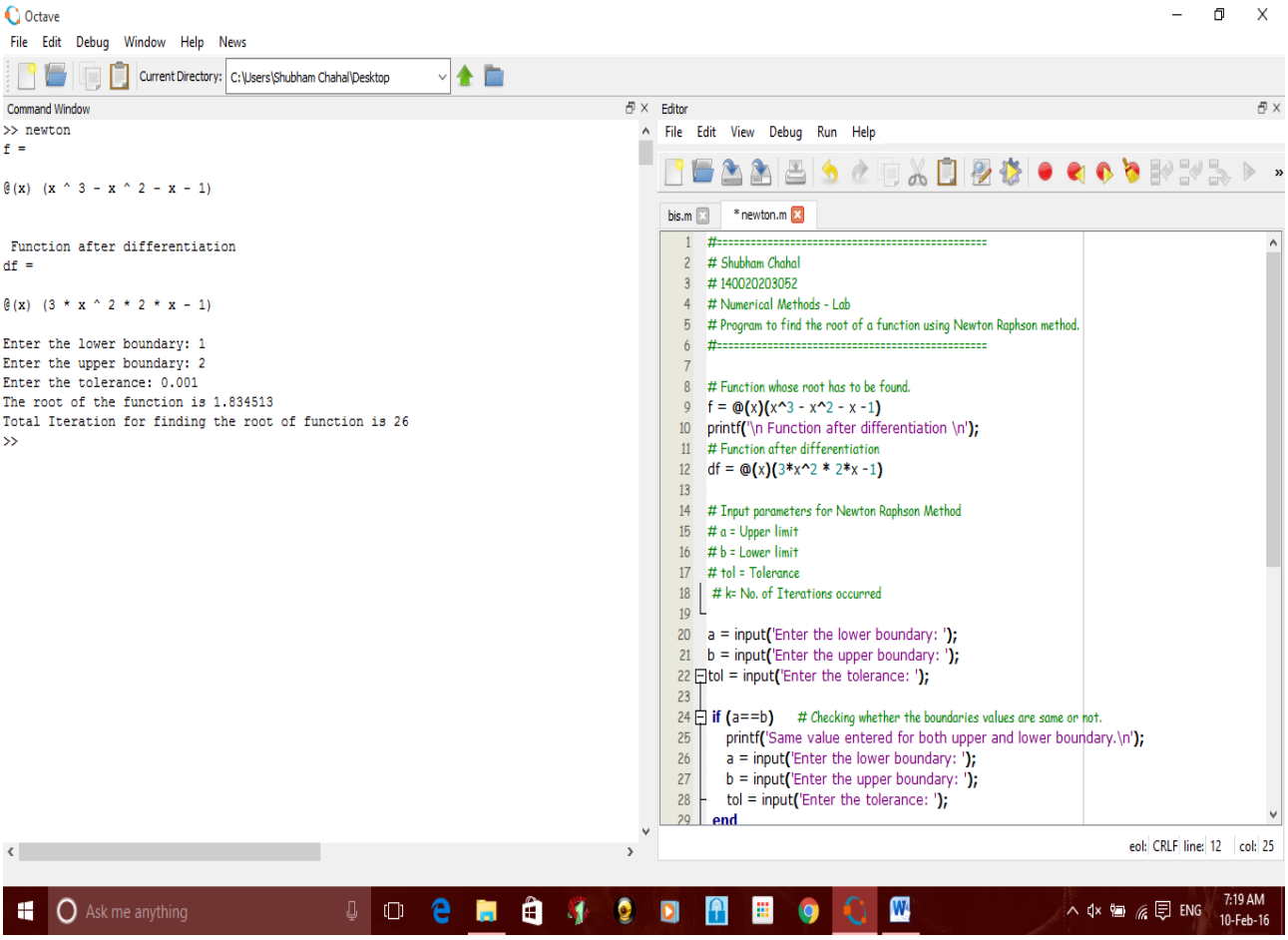
```
# Newton Raphson Method code

k=0;    # Counter for the no. of iterations

while(abs(b-a)>tol)
    y=a-f(a)/df(a);
    b=a;
    a=y;
    k++;
end

# Display the value of a, i.e root of the function.
printf('The root of the function is %f \n',a);

# Display the value of k,, i.e the number of iterations occurred.
printf('Total Iteration for finding the root of function is %d\n',k);
```

Outputs:

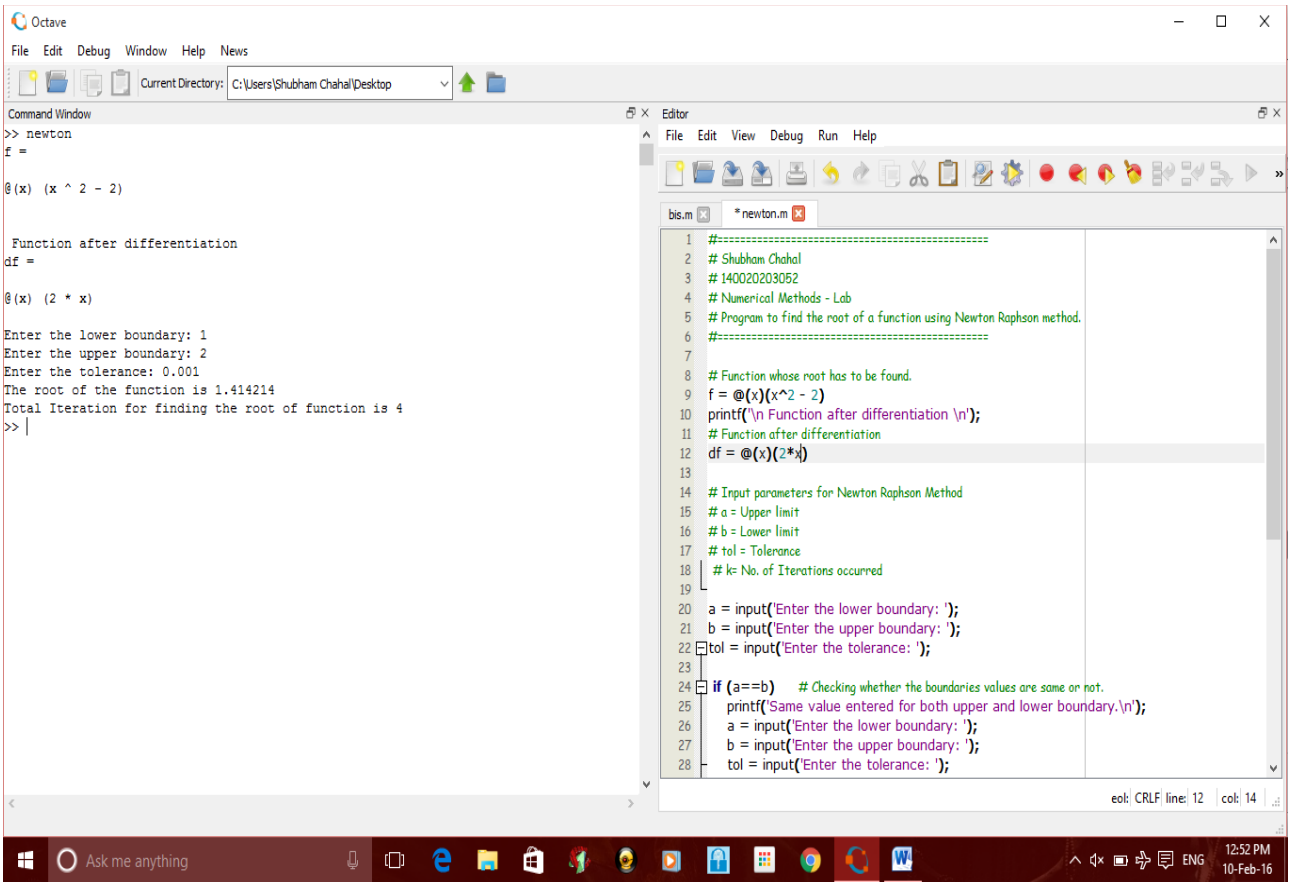
a). f(x) = cos(x) – x.e^x



b). $f(x) = x^3 - x^2 - x - 1$


Octave
File Edit Debug Window Help News
Current Directory: C:\Users\Shubham Chahal\Desktop
Command Window
>> newton
f =
@(x) (x ^ 3 - x ^ 2 - x - 1)
Function after differentiation
df =
@(x) (3 * x ^ 2 * 2 * x - 1)
Enter the lower boundary: 1
Enter the upper boundary: 2
Enter the tolerance: 0.001
The root of the function is 1.834513
Total Iteration for finding the root of function is 26
>>
Editor
File Edit View Debug Run Help
bis.m
*newton.m
#==========
Shubham Chahal
140020203052
Numerical Methods - Lab
Program to find the root of a function using Newton Raphson method.
#==========
# Function whose root has to be found.
f = @(x)(x^3 - x^2 - x -1)
printf("\n Function after differentiation \n");
# Function after differentiation
df = @(x)(3*x^2 * 2*x -1)
# Input parameters for Newton Raphson Method
# a = Upper limit
# b = Lower limit
# tol = Tolerance
# k= No. of Iterations occurred
a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');
if (a==b) # Checking whether the boundaries values are same or not.
printf('Same value entered for both upper and lower boundary.\n');
a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');
end
eol: CRLF line: 12 col: 25
Ask me anything
ENG
7:19 AM
10-Feb-16

c). $f(x) = x^2 - 2$

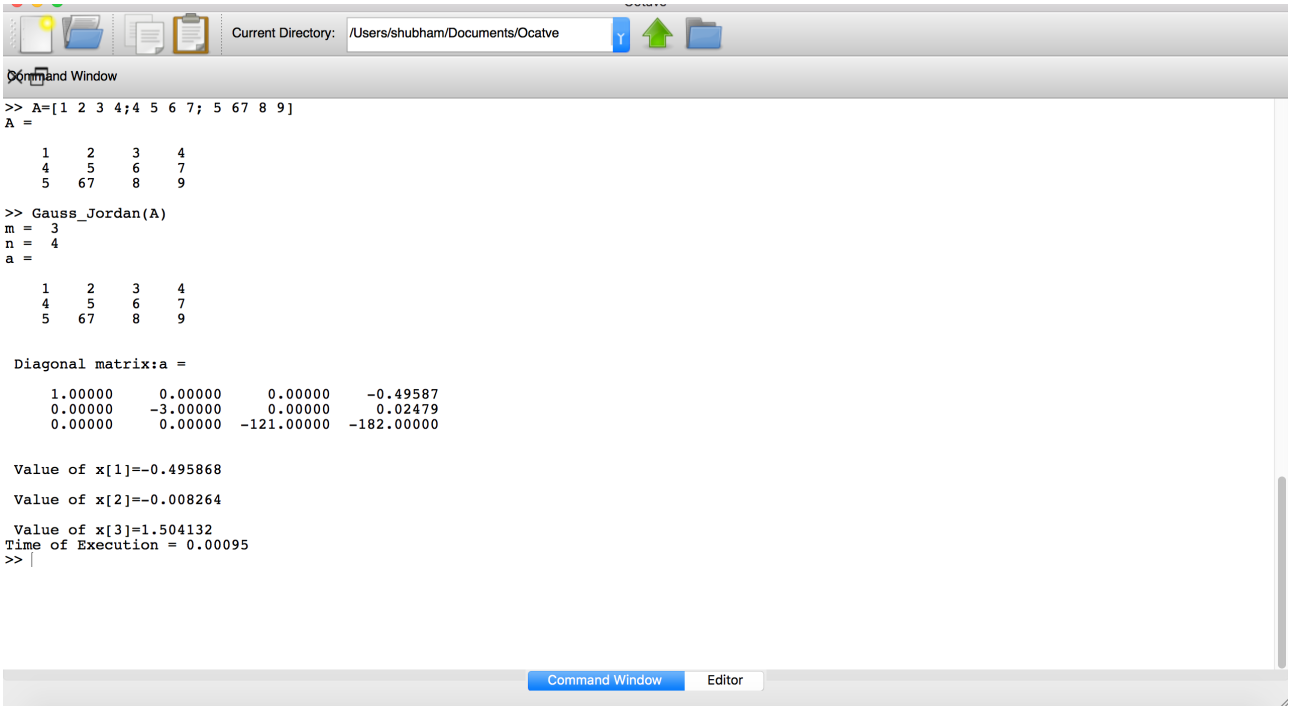

Octave
File Edit Debug Window Help News
Current Directory: C:\Users\Shubham Chahal\Desktop
Command Window
>> newton
f =
@(x) (x ^ 2 - 2)
Function after differentiation
df =
@(x) (2 * x)
Enter the lower boundary: 1
Enter the upper boundary: 2
Enter the tolerance: 0.001
The root of the function is 1.414214
Total Iteration for finding the root of function is 4
>>
Editor
File Edit View Debug Run Help
bis.m
*newton.m
#==========
Shubham Chahal
140020203052
Numerical Methods - Lab
Program to find the root of a function using Newton Raphson method.
#==========
# Function whose root has to be found.
f = @(x)(x^2 - 2)
printf('\n Function after differentiation \n');
# Function after differentiation
df = @(x)(2*x)
# Input parameters for Newton Raphson Method
# a = Upper limit
# b = Lower limit
# tol = Tolerance
# k= No. of Iterations occurred
a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');
if (a==b) # Checking whether the boundaries values are same or not.
printf('Same value entered for both upper and lower boundary.\n');
a = input('Enter the lower boundary: ');
b = input('Enter the upper boundary: ');
tol = input('Enter the tolerance: ');
eol: CRLF line: 12 col: 14
Ask me anything
ENG
12:52 PM
10-Feb-16

Write a Program to implement Gauss-Jordan Method.

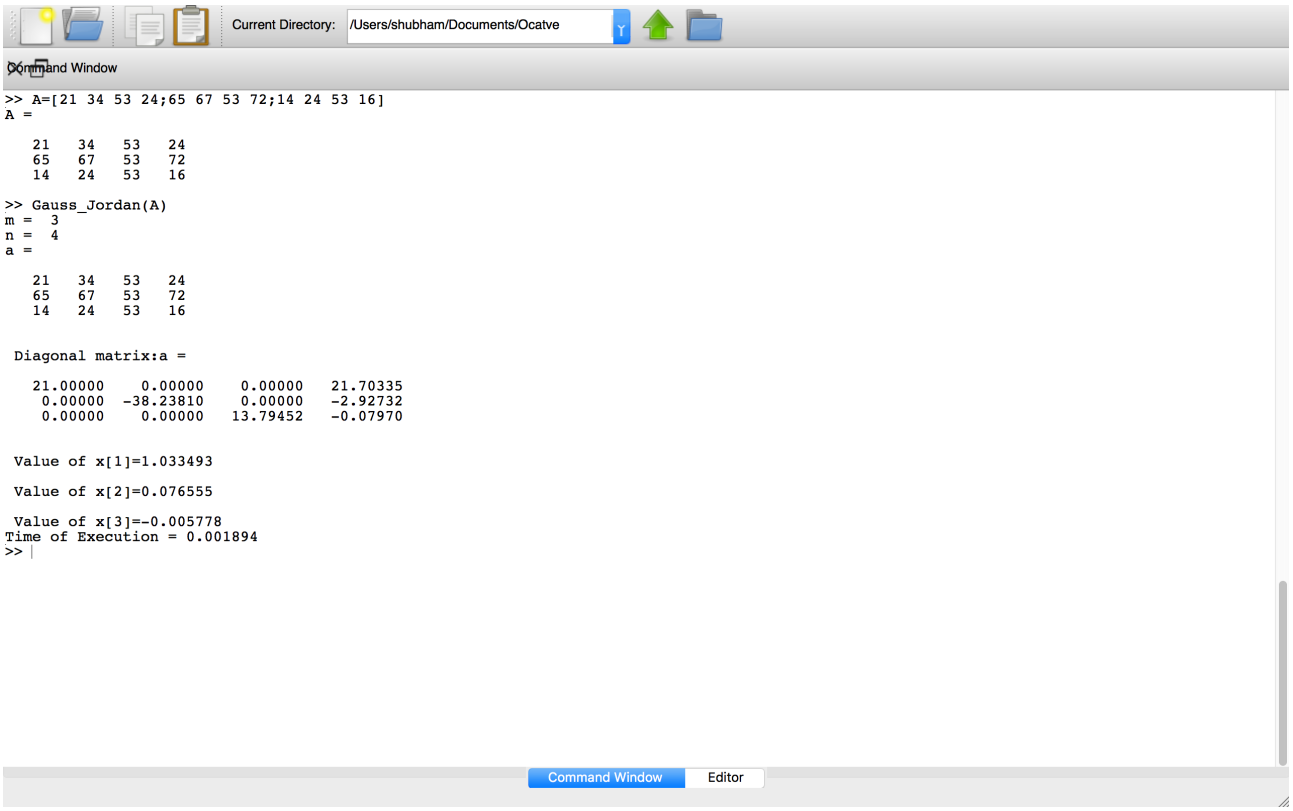
```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to solve a system of equations using Gauss-Jordan Method.

#=========================================================================


function Gauss_Jordan(a)

 t=cputime;

 [m,n]=size(a)

 a

 for (j=1:m)

   for (i=1:m)

     if(i!=j)

       c=a(i,j)/a(j,j);

     for (k=1:n)

       a(i,k)=a(i,k)-c*a(j,k);

     end

     end

  end

 end

 fprintf("\n Diagonal matrix:");

 a

 for(i=1:m)

   x=a(i,n)/a(i,i);
   fprintf("\n Value of x[%d]=%f\n",i,x);
 end
```

```
  t=cputime-t;
  fprintf("Time of Execution = %d\n",t);
endfunction
```

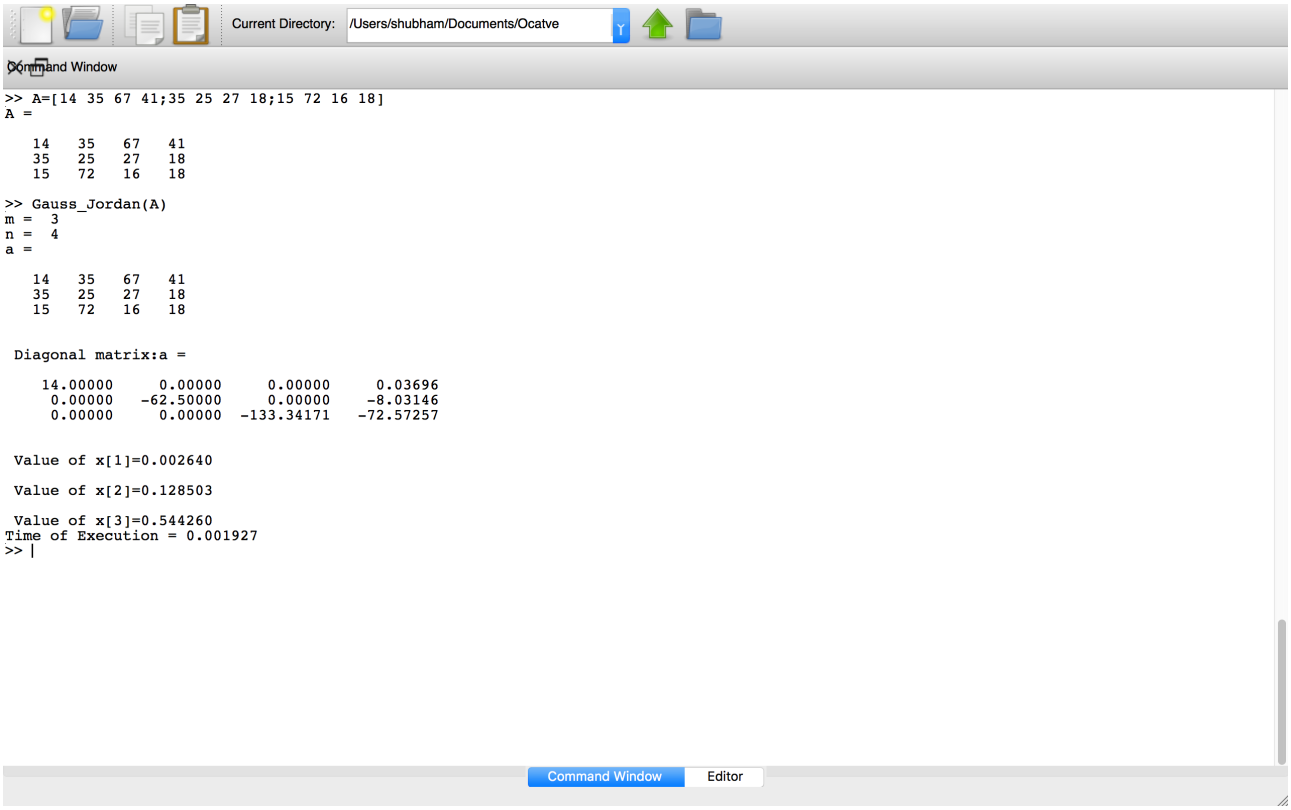
Outputs:
(a).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[1 2 3 4;4 5 6 7; 5 67 8 9]
A =
1 2 3 4
4 5 6 7
5 67 8 9
>> Gauss_Jordan(A)
m = 3
n = 4
a =
1 2 3 4
4 5 6 7
5 67 8 9
Diagonal matrix:a =
1.00000 0.00000 0.00000 -0.49587
0.00000 -3.00000 0.00000 0.02479
0.00000 0.00000 -121.00000 -182.00000
Value of x[1]=-0.495868
Value of x[2]=-0.008264
Value of x[3]=1.504132
Time of Execution = 0.00095
>>
Command Window
Editor

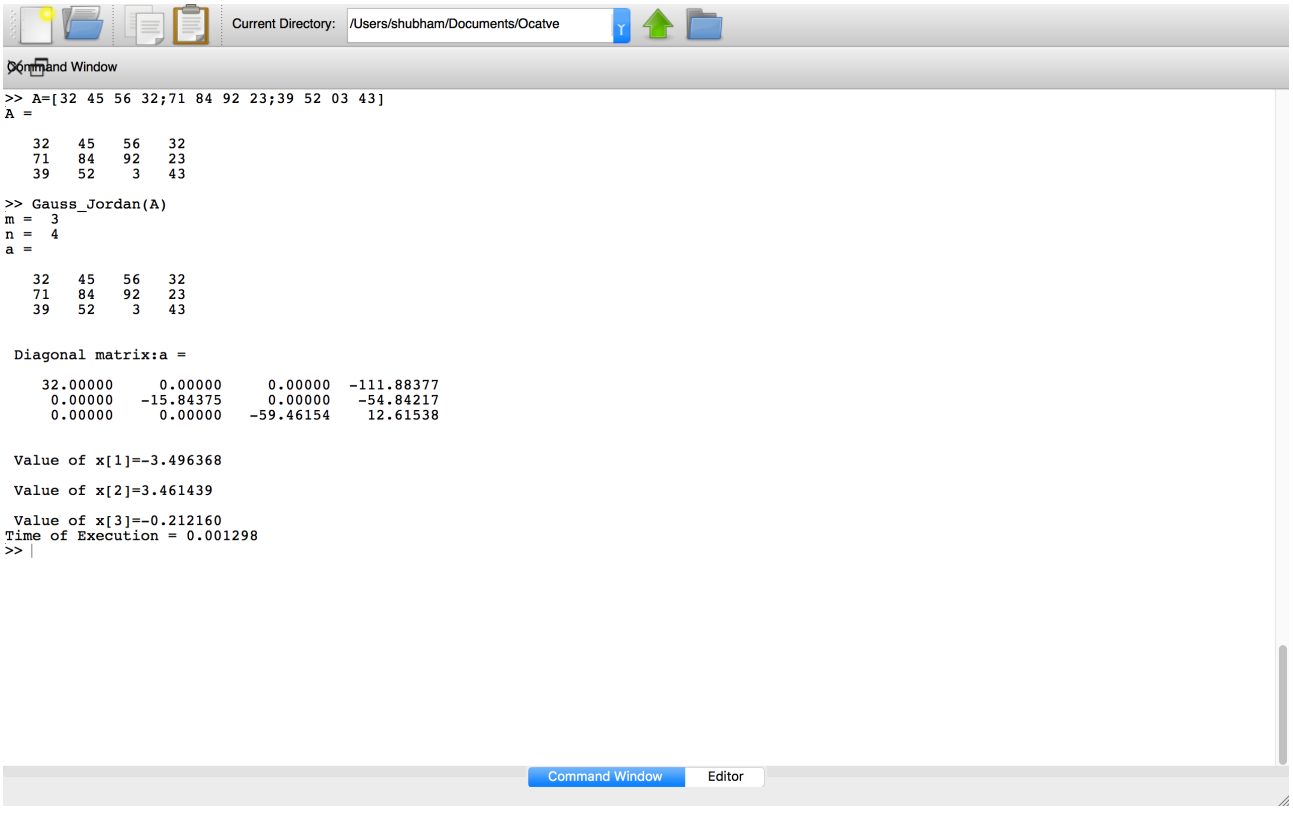
(b).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[21 34 53 24;65 67 53 72;14 24 53 16]
A =
21 34 53 24
65 67 53 72
14 24 53 16
>> Gauss_Jordan(A)
m = 3
n = 4
a =
21 34 53 24
65 67 53 72
14 24 53 16
Diagonal matrix:a =
21.00000 0.00000 0.00000 21.70335
0.00000 -38.23810 0.00000 -2.92732
0.00000 0.00000 13.79452 -0.07970
Value of x[1]=1.033493
Value of x[2]=0.076555
Value of x[3]=-0.005778
Time of Execution = 0.001894
>>
Command Window
Editor

(c).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[14 35 67 41;35 25 27 18;15 72 16 18]
A =
14 35 67 41
35 25 27 18
15 72 16 18
>> Gauss_Jordan(A)
m = 3
n = 4
a =
14 35 67 41
35 25 27 18
15 72 16 18
Diagonal matrix:a =
14.00000 0.00000 0.00000 0.03696
0.00000 -62.50000 0.00000 -8.03146
0.00000 0.00000 -133.34171 -72.57257
Value of x[1]=0.002640
Value of x[2]=0.128503
Value of x[3]=0.544260
Time of Execution = 0.001927
>>
Command Window
Editor

(d).

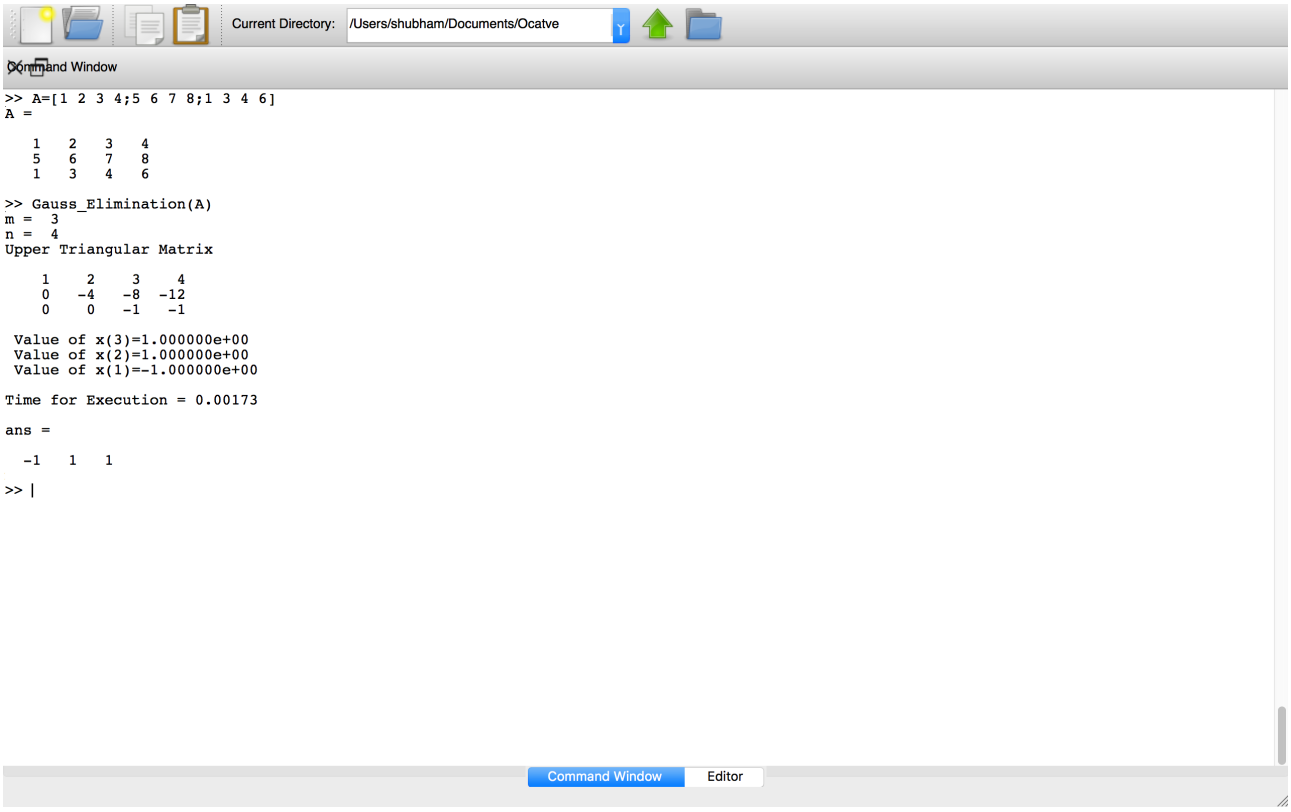

Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[32 45 56 32;71 84 92 23;39 52 03 43]
A =
32 45 56 32
71 84 92 23
39 52 3 43
>> Gauss_Jordan(A)
m = 3
n = 4
a =
32 45 56 32
71 84 92 23
39 52 3 43
Diagonal matrix:a =
32.00000 0.00000 0.00000 -111.88377
0.00000 -15.84375 0.00000 -54.84217
0.00000 0.00000 -59.46154 12.61538
Value of x[1]=-3.496368
Value of x[2]=3.461439
Value of x[3]=-0.212160
Time of Execution = 0.001298
>>
Command Window
Editor

Write a Program to implement Gauss-Elimination Method.

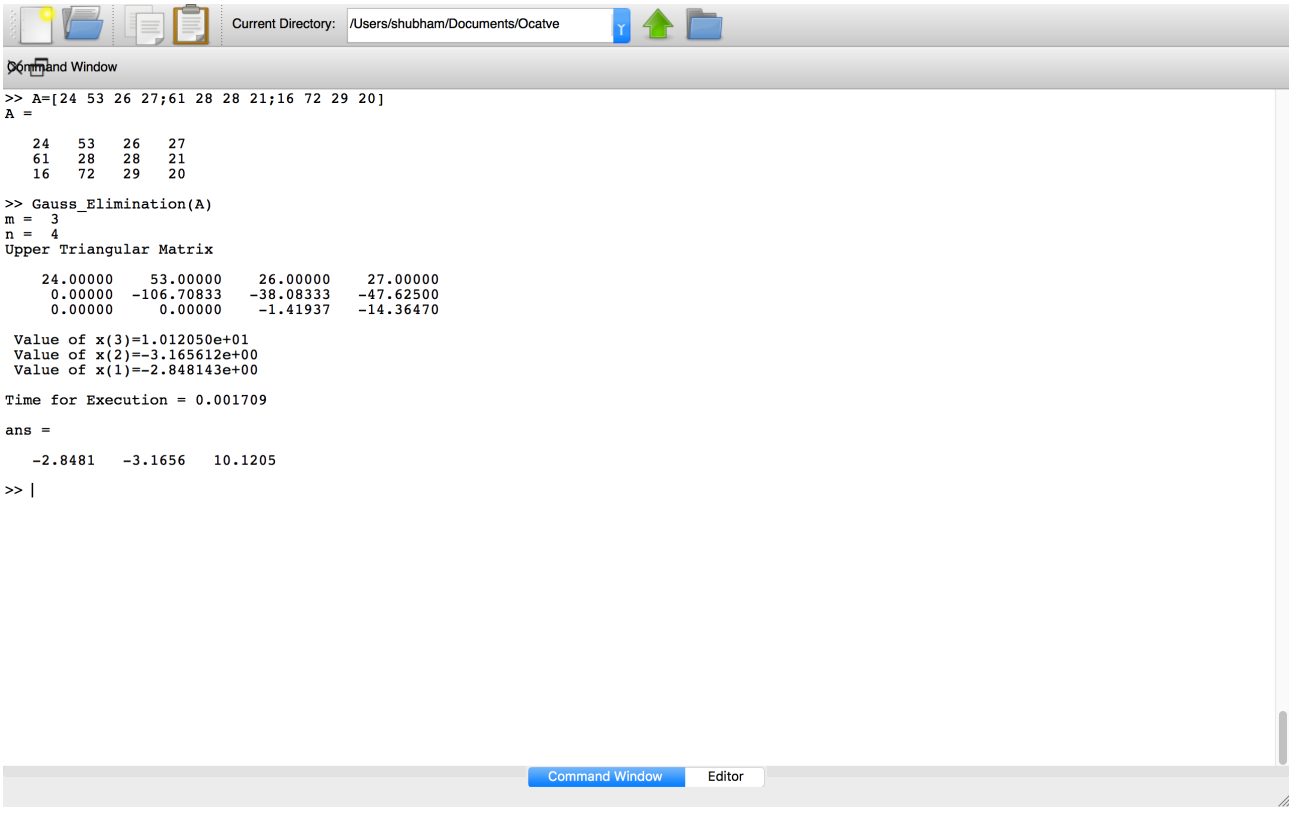
```
#=============================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to solve a system of equations using Gauss-Elimination Method.

#=============================================================================

function [x]= Gauss_Elimination(a)
  t=cputime;
  [m,n]=size(a)
  for j=1:m-1
    for i=j+1:m
       if(i!=j)
          c=a(i,j)/a(j,j);
          for k=j:m+1
              a(i,k)=a(i,k)-(c*a(j,k));
          end
       end
    end
  end
  display('Upper Triangular Matrix')
  display(a)
  for i=m:-1:1
      s=0;
       for j=i+1:m
          s+=a(i,j)*x(j);
       end
```

```
    x(i)=(a(i,m+1)-s)/a(i,i);
    fprintf(' Value of x(%d)=%e \n',i,x(i));

  end
  t=cputime-t;
    fprintf("\nTime for Execution = %d\n\n",t);

endfunction
```

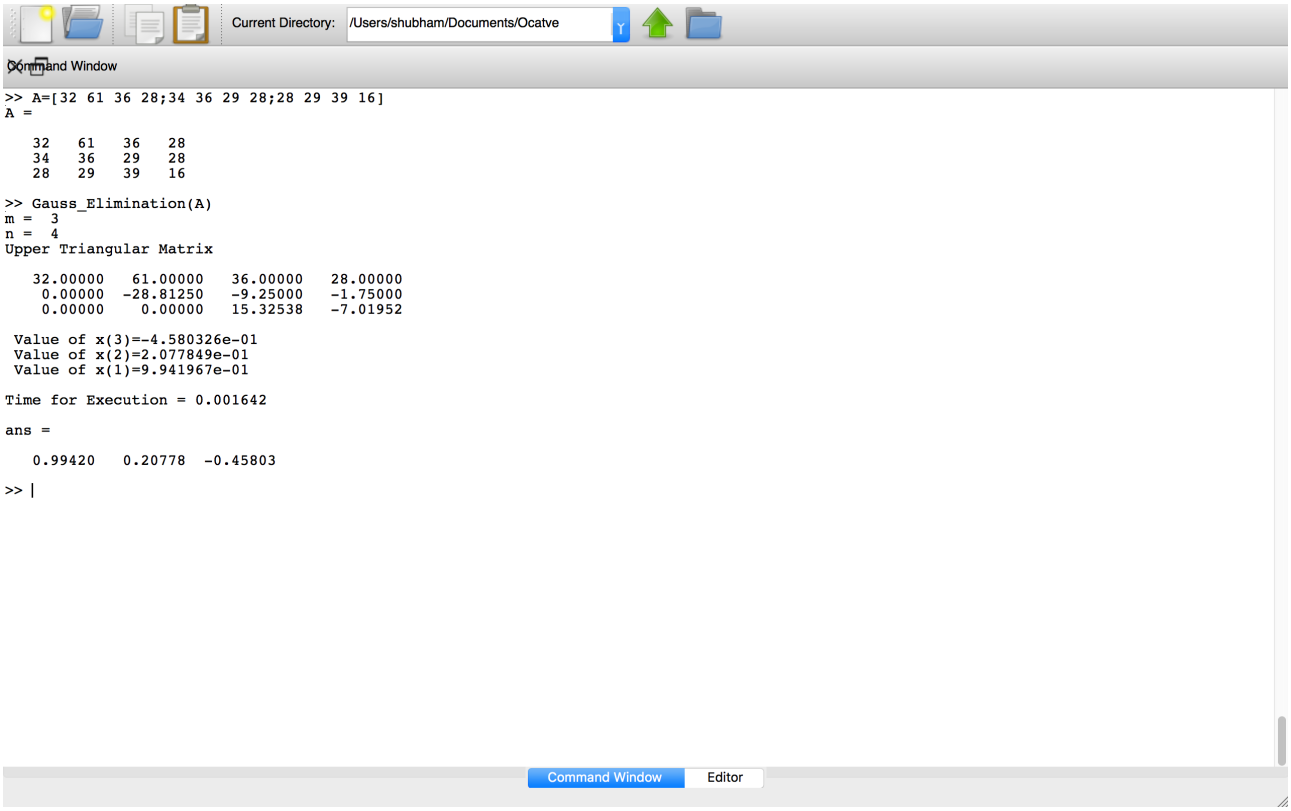
Outputs:
(a).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[1 2 3 4;5 6 7 8;1 3 4 6]
A =
1 2 3 4
5 6 7 8
1 3 4 6
>> Gauss_Elimination(A)
m = 3
n = 4
Upper Triangular Matrix
1 2 3 4
0 -4 -8 -12
0 0 -1 -1
Value of x(3)=1.000000e+00
Value of x(2)=1.000000e+00
Value of x(1)=-1.000000e+00
Time for Execution = 0.00173
ans =
-1 1 1
>>
Command Window
Editor

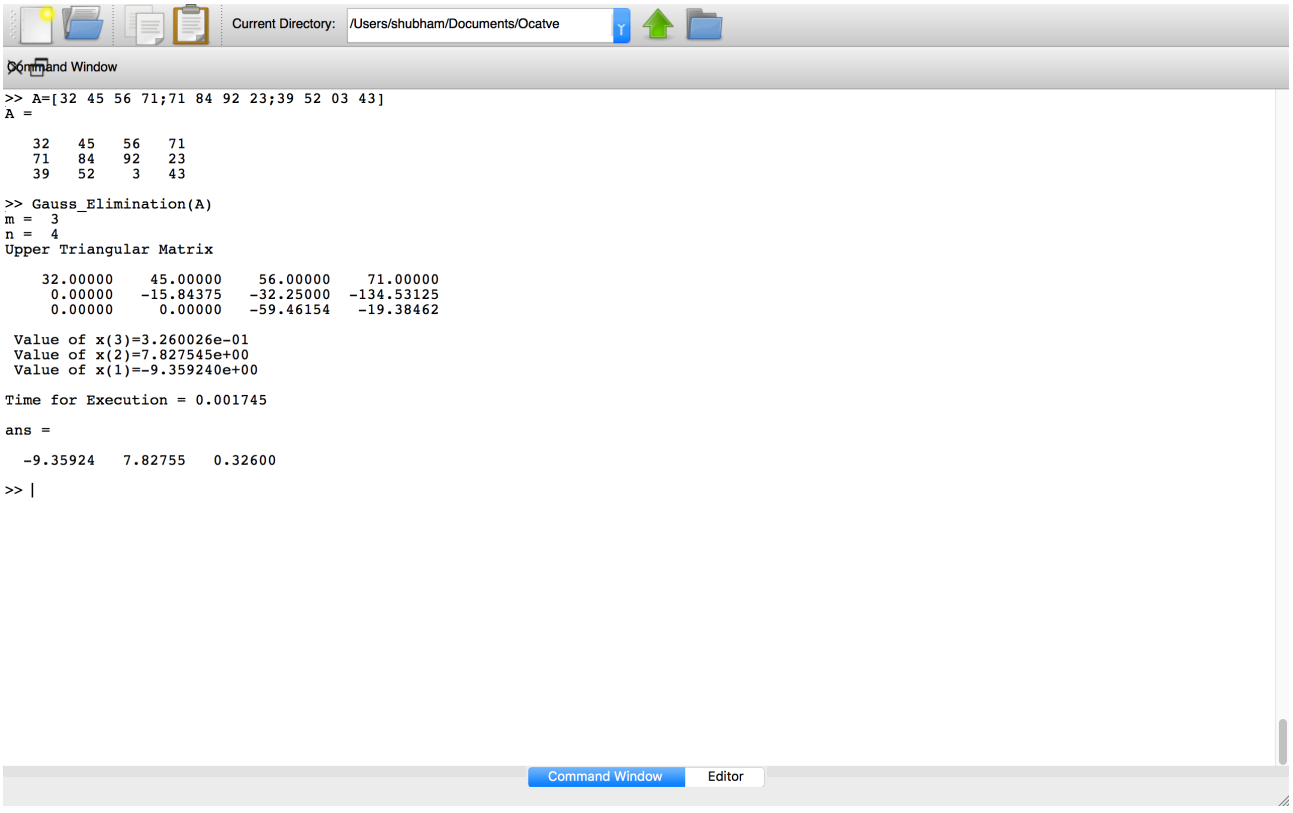
(b).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[24 53 26 27;61 28 28 21;16 72 29 20]
A =
24 53 26 27
61 28 28 21
16 72 29 20
>> Gauss_Elimination(A)
m = 3
n = 4
Upper Triangular Matrix
24.00000 53.00000 26.00000 27.00000
0.00000 -106.70833 -38.08333 -47.62500
0.00000 0.00000 -1.41937 -14.36470
Value of x(3)=1.012050e+01
Value of x(2)=-3.165612e+00
Value of x(1)=-2.848143e+00
Time for Execution = 0.001709
ans =
-2.8481 -3.1656 10.1205
>>
Command Window
Editor

(c).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[32 61 36 28;34 36 29 28;28 29 39 16]
A =
32 61 36 28
34 36 29 28
28 29 39 16
>> Gauss_Elimination(A)
m = 3
n = 4
Upper Triangular Matrix
32.00000 61.00000 36.00000 28.00000
0.00000 -28.81250 -9.25000 -1.75000
0.00000 0.00000 15.32538 -7.01952
Value of x(3)=-4.580326e-01
Value of x(2)=2.077849e-01
Value of x(1)=9.941967e-01
Time for Execution = 0.001642
ans =
0.99420 0.20778 -0.45803
>>
Command Window
Editor

(d).

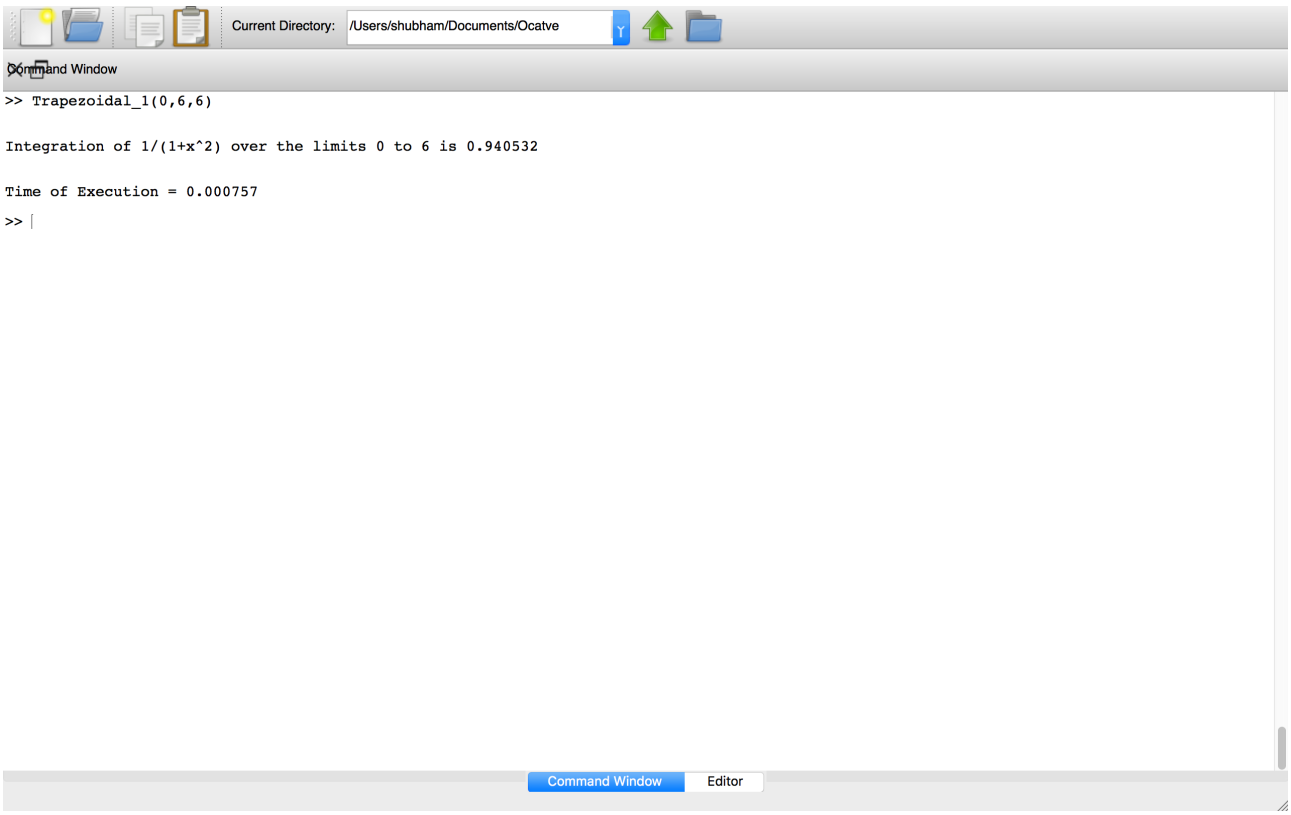

Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> A=[32 45 56 71;71 84 92 23;39 52 03 43]
A =
32 45 56 71
71 84 92 23
39 52 3 43
>> Gauss_Elimination(A)
m = 3
n = 4
Upper Triangular Matrix
32.00000 45.00000 56.00000 71.00000
0.00000 -15.84375 -32.25000 -134.53125
0.00000 0.00000 -59.46154 -19.38462
Value of x(3)=3.260026e-01
Value of x(2)=7.827545e+00
Value of x(1)=-9.359240e+00
Time for Execution = 0.001745
ans =
-9.35924 7.82755 0.32600
>>
Command Window
Editor

Write a Program to implement Trapezoidal Method

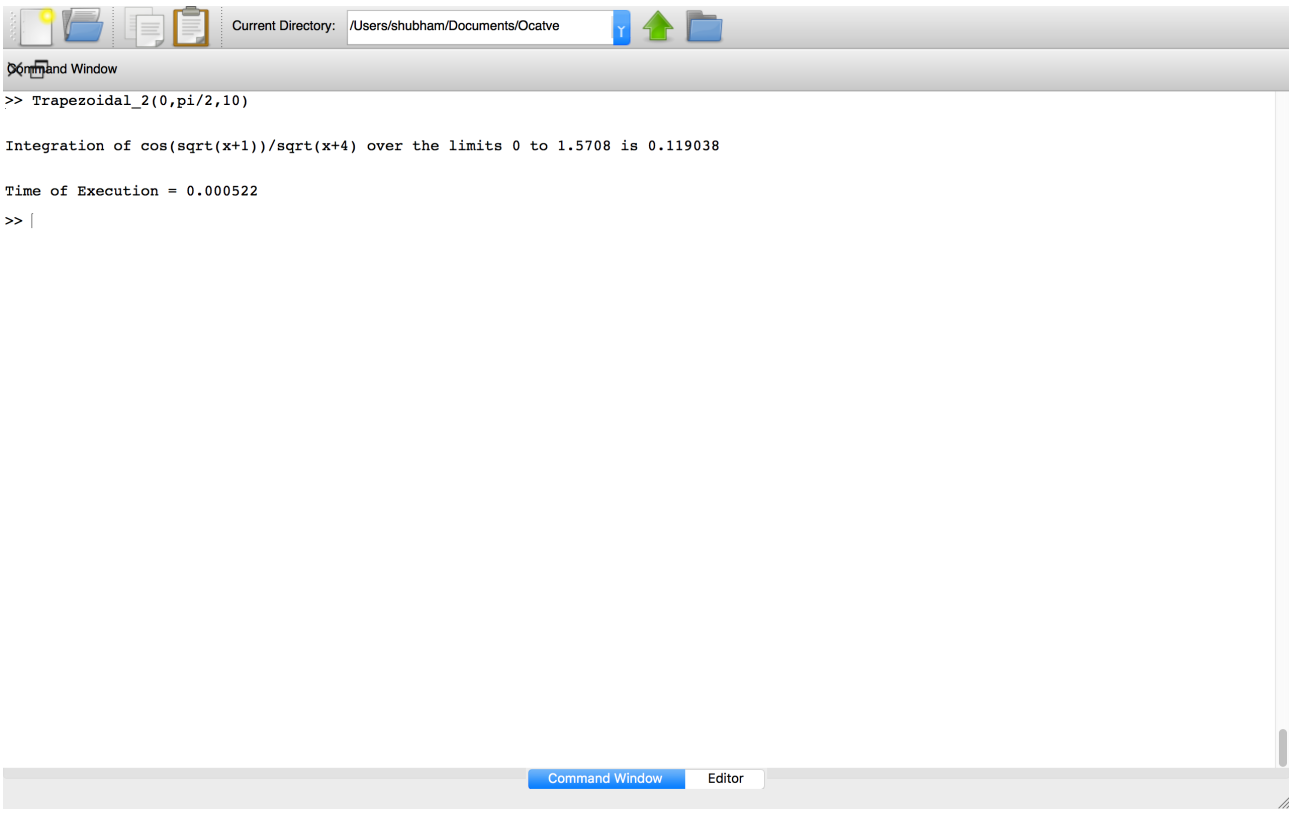
```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to integrate an equation using Trapezoidal Method.

#=========================================================================

function Trapezoidal_1( a, b, n)

  t=cputime;

  h = (b - a)/ n;

  s = f(a) + f(a + n*h);

  for i = 1 : n-1

    s = s + 2*f(a + i*h);

  end

  fprintf("\n\nIntegration of 1/(1+x^2) over the limits %d to %d is %d\n ", a,b,s*h/3);

  t=cputime-t;

 fprintf("\n\nTime of Execution = %d\n\n",t);

end

function f = f(x)

  f = 1/(1+x*x);

end
```

```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to integrate an equation using Trapezoidal Method.

#=========================================================================

function Trapezoidal_2( a, b, n)

  t=cputime;

  h = (b - a)/ n;

  s = f(a) + f(a + n*h);

  for i = 1 : n-1

    s = s + 2*f(a + i*h);

  end

  fprintf("\n\nIntegration of cos(sqrt(x+1))/sqrt(x+4) over the limits %d to %d is %d\n ", a,b,s*h/3);

  t=cputime-t;

 fprintf("\n\nTime of Execution = %d\n\n",t);

end

function f = f(x)

  f = cos(sqrt(x+1))/sqrt(x+4);

end
```

Outputs:
(a).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Trapezoidal_1(0,6,6)
Integration of 1/(1+x^2) over the limits 0 to 6 is 0.940532
Time of Execution = 0.000757
>>
Command Window
Editor

(b).

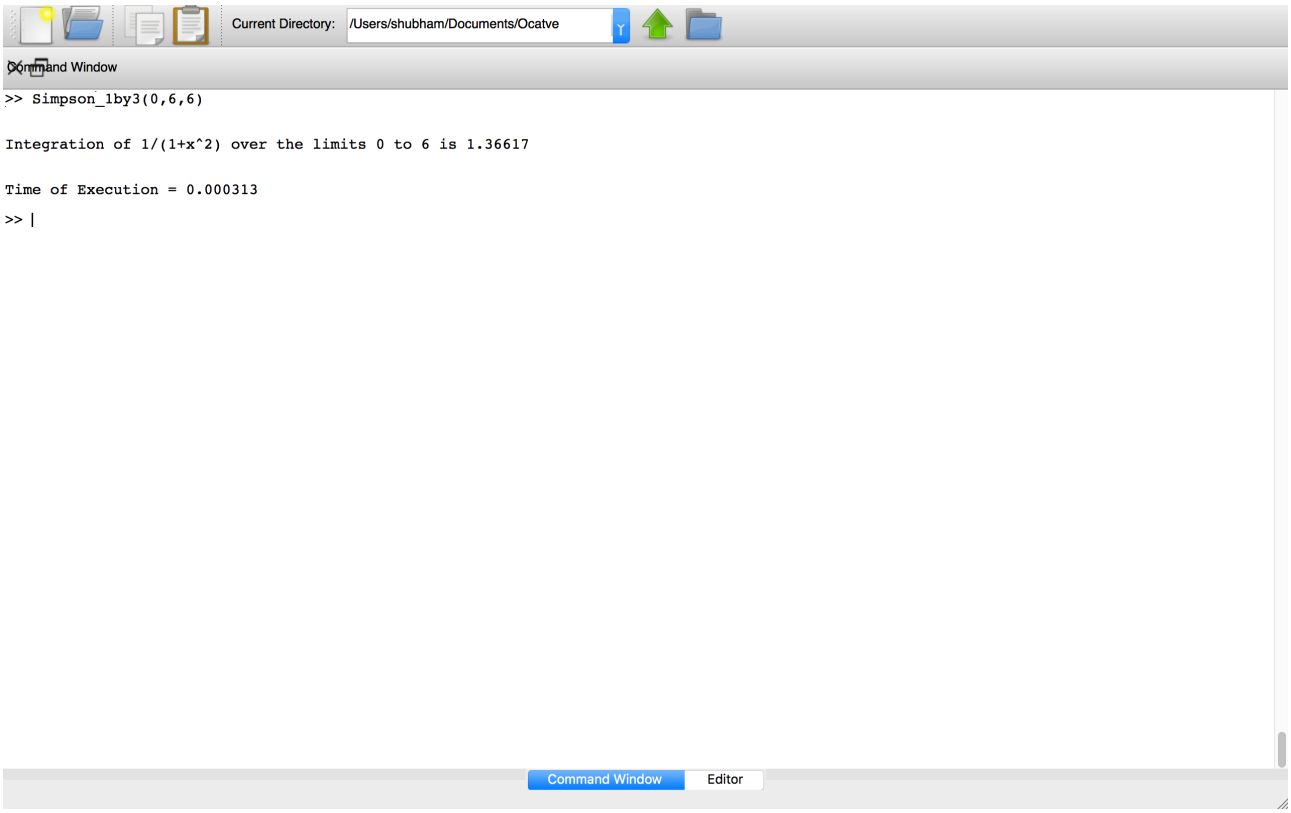

Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Trapezoidal_2(0,pi/2,10)
Integration of cos(sqrt(x+1))/sqrt(x+4) over the limits 0 to 1.5708 is 0.119038
Time of Execution = 0.000522
>>
Command Window
Editor

Write a Program to implement Simpson’s 1/3 Method

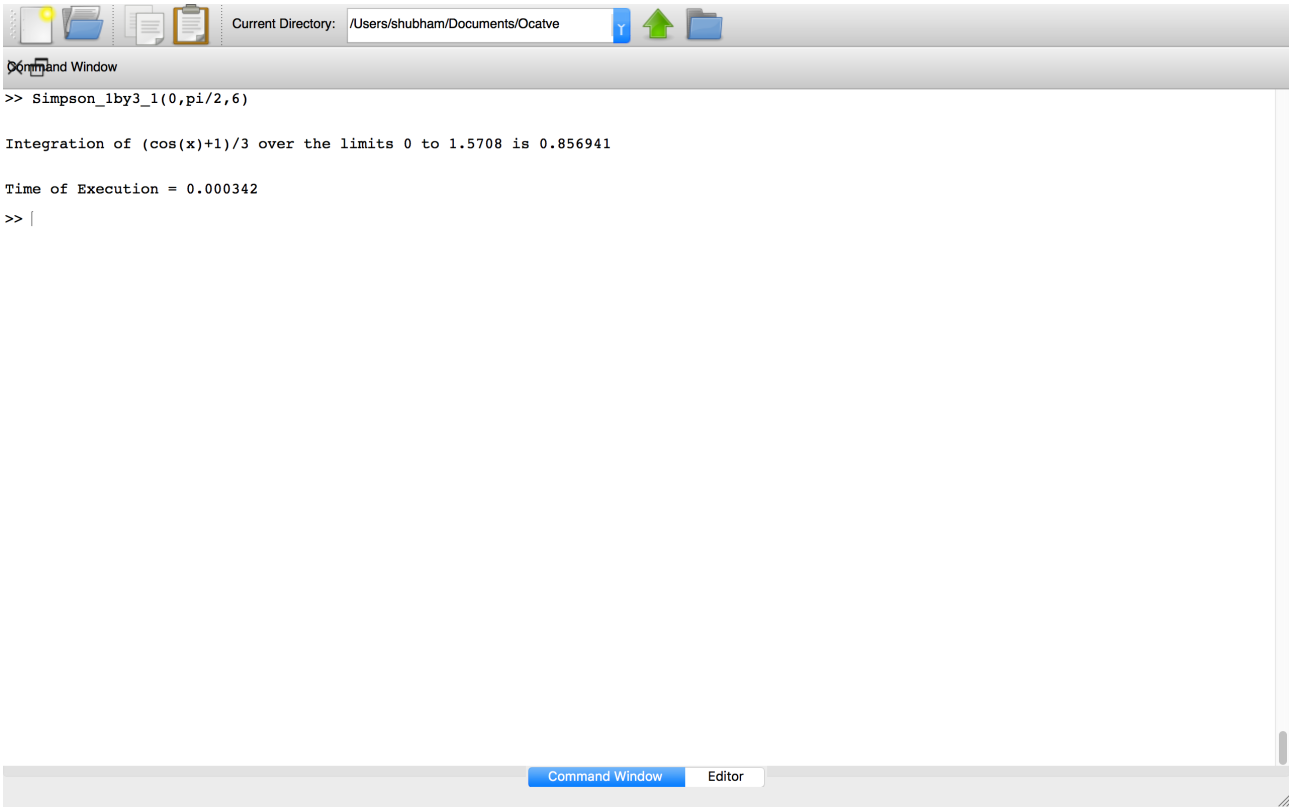
```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to integrate an equation using Simpson's 1/3 Method.

#=========================================================================


function Simpson_1by3( a, b, n)

 t=cputime;

 h = (b - a)/ n;

 s = f(a) + f(a + n*h);

 for i = 1 : n-1

   if mod(i,2) == 0

      s = s + 2*f(a+i*h);

   else

      s = s + 4*f(a+i*h);

   endif

 end

 fprintf("\n\nIntegration of 1/(1+x^2) over the limits %d to %d is %d\n ", a,b,s*h/3);

 t=cputime-t;

 fprintf("\n\nTime of Execution = %d\n\n",t);

end

function f = f(x)

  f = 1/(1+x*x);

end
```

```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to integrate an equation using Simpson's 1/3 Method.

#=========================================================================

function Simpson_1by3_1( a, b, n)

 t=cputime;

 h = (b - a)/ n;

 s = f(a) + f(a + n*h);

 for i = 1 : n-1

   if mod(i,2) == 0

      s = s + 2*f(a+i*h);

   else

      s = s + 4*f(a+i*h);

   endif

 end

 fprintf("\n\nIntegration of (cos(x)+1)/3 over the limits %d to %d is %d\n ", a,b,s*h/3);

 t=cputime-t;

 fprintf("\n\nTime of Execution = %d\n\n",t);

end

function f = f(x)

  f = (cos(x)+1)/3;

end
```

Outputs:
(a).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Simpson_1by3(0,6,6)
Integration of 1/(1+x^2) over the limits 0 to 6 is 1.36617
Time of Execution = 0.000313
>>
Command Window
Editor

(b).

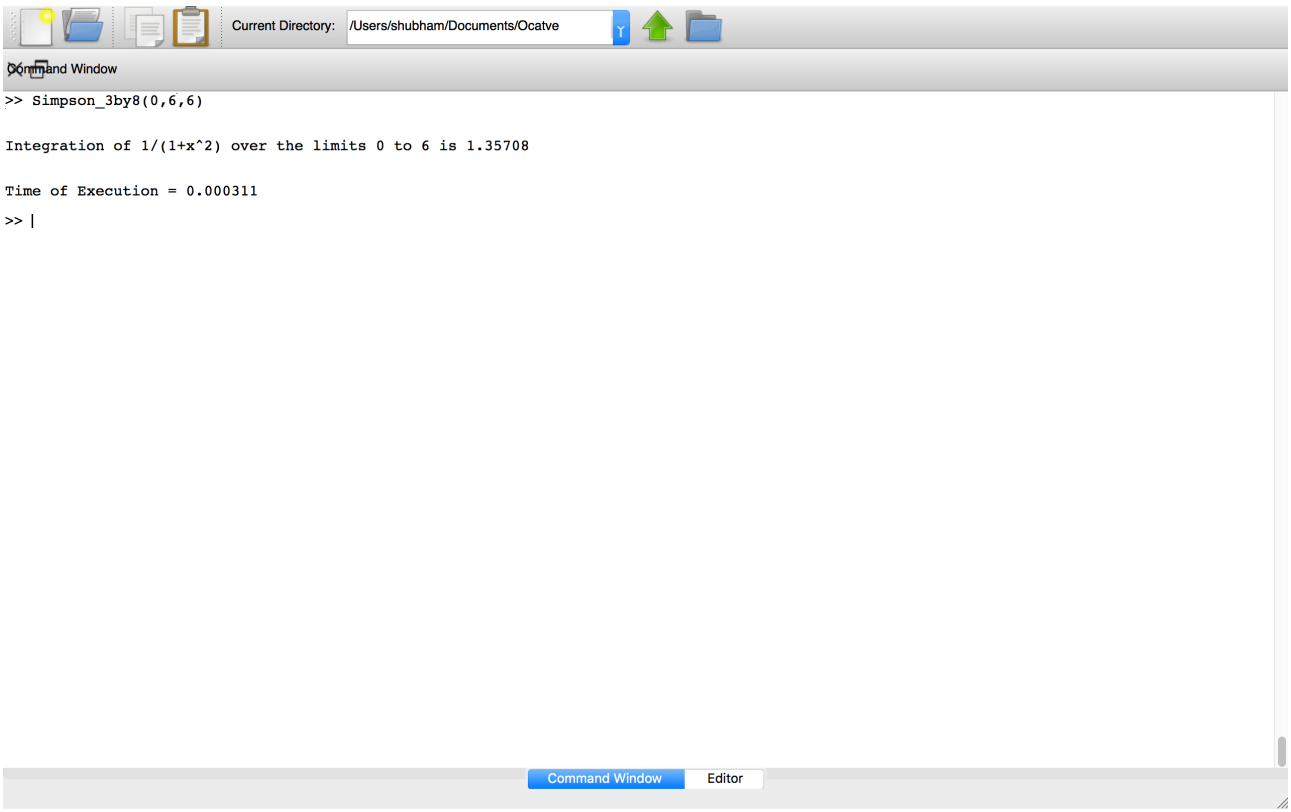

Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Simpson_1by3_1(0,pi/2,6)
Integration of (cos(x)+1)/3 over the limits 0 to 1.5708 is 0.856941
Time of Execution = 0.000342
>>
Command Window
Editor

Write a Program to implement Simpson’s 3/8 Method

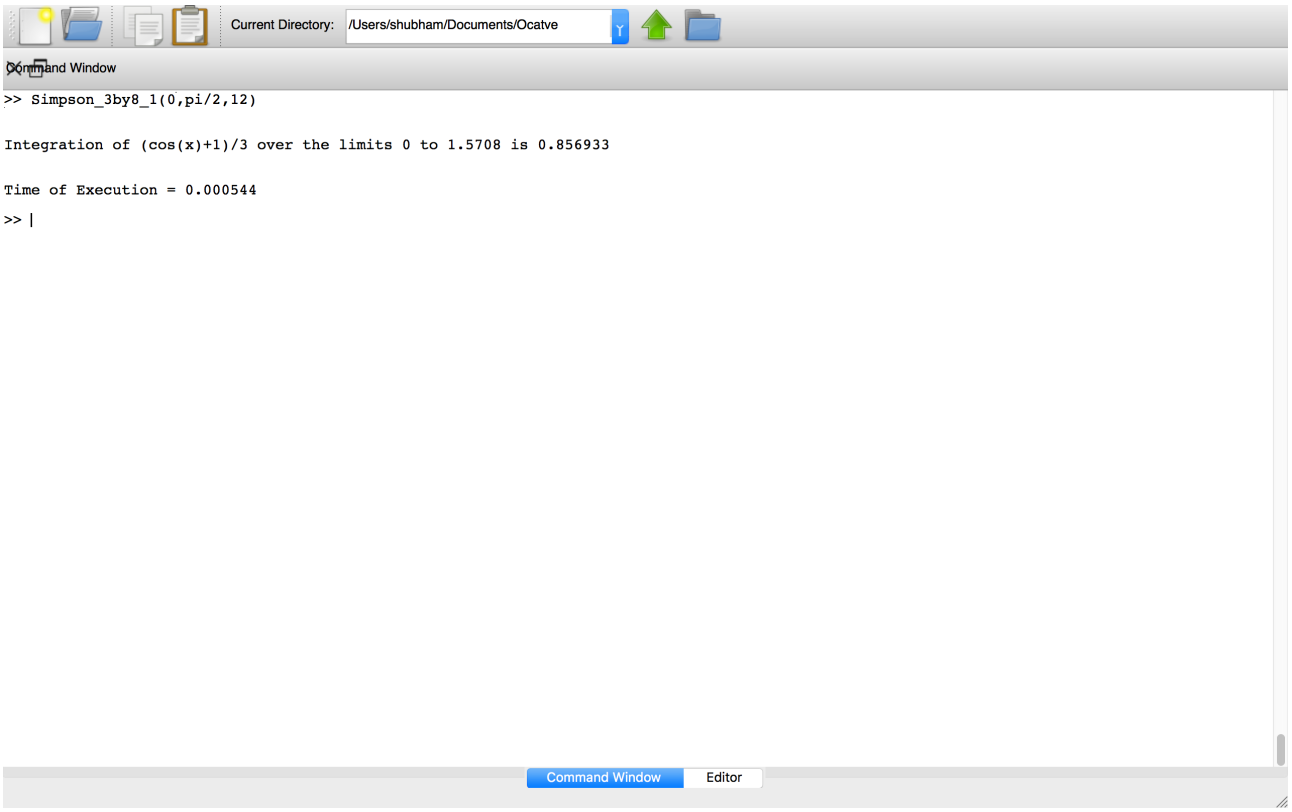
```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to integrate an equation using Simpson's 3/8 Method.

#=========================================================================

function Simpson_3by8( a, b, n)

  t=cputime;

  h = (b - a)/ n;

  s = f(a) + f(a + n*h);

  for i = 1 : n-1

     if mod(i,3) == 0

         s = s + 2*f(a+i*h);

    else

         s = s + 3*f(a+i*h);

    endif

  end

  fprintf("\n\nIntegration of 1/(1+x^2) over the limits %d to %d is %d\n",a,b,s*h*3/8);

  t=cputime-t;

  fprintf("\n\nTime of Execution = %d\n\n",t);

end

function f = f(x)
  f = 1/(1+x*x);
end
```

```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to integrate an equation using Simpson's 3/8 Method.

#=========================================================================


function Simpson_3by8_1( a, b, n)

  t=cputime;

  h = (b - a)/ n;

  s = f(a) + f(a + n*h);

  for i = 1 : n-1

     if mod(i,3) == 0

         s = s + 2*f(a+i*h);

    else

         s = s + 3*f(a+i*h);

    endif

  end

  fprintf("\n\nIntegration of (cos(x)+1)/3 over the limits %d to %d is %d\n",a,b,s*h*3/8);

  t=cputime-t;

  fprintf("\n\nTime of Execution = %d\n\n",t);

end

function f = f(x)
  f = (cos(x)+1)/3;
end
```

Outputs:
(a).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Simpson_3by8(0,6,6)
Integration of 1/(1+x^2) over the limits 0 to 6 is 1.35708
Time of Execution = 0.000311
>> |
Command Window
Editor

(b).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Simpson_3by8_1(0,pi/2,12)
Integration of (cos(x)+1)/3 over the limits 0 to 1.5708 is 0.856933
Time of Execution = 0.000544
>> |
Command Window
Editor

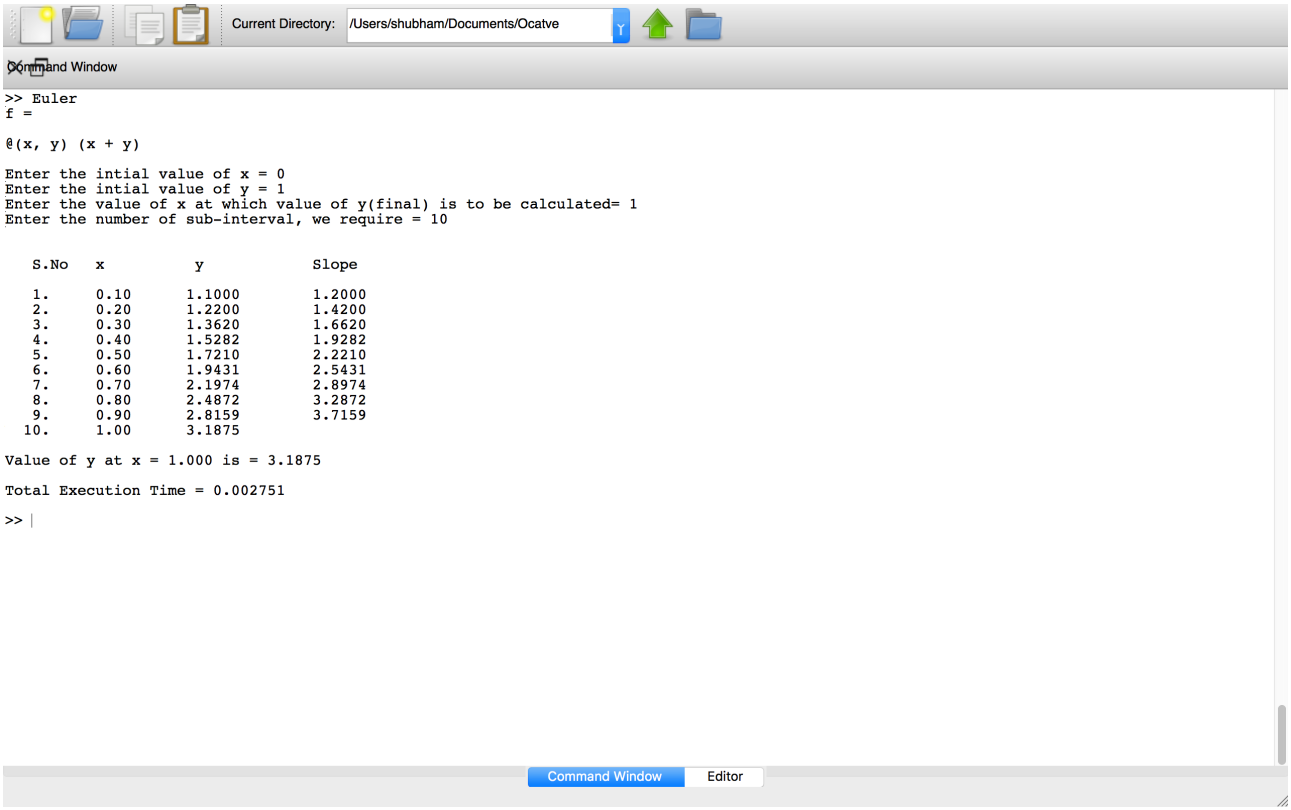
Write a Program to implement Euler’s Method

```
#=========================================================================
# Shubham Chahal
# 140020203052
# Numerical Methods - Lab
# Program to obtain solution of an equation using Euler's Method.
#=========================================================================

f=@(x,y)(x+y)
    x0=input('Enter the intial value of x = ');
    y0=input('Enter the intial value of y = ');
    xf=input('Enter the value of x at which value of y(final) is to be calculated= ');
    n=input('Enter the number of sub-interval, we require = ');
    t=cputime;
    x=x0;
    y=y0;
    h=(xf-x0)/n;

    fprintf('\n\n   S.No   x          y          Slope       \n\n');
    k=0;

# Euler's Method Code

    for  i=1:n
        k++;
        y=y+h*f(x,y);
        x=x+h;
        if(k<10)&&(k!=n)
            fprintf("   %d.     %.2f      %.4f       %.4f\n",k,x,y,f(x,y));
```

```
        elseif(k==n)&&(k<10)
                fprintf("   %d.     %.2f      %.4f        \n",k,x,y);
        elseif(k==n)&&(k>9)
                fprintf("  %d.     %.2f      %.4f        \n",k,x,y);
        else
                fprintf("  %d.     %.2f      %.4f        %.4f\n",k,x,y,f(x,y));
        end
end
fprintf('\nValue of y at x = %.3f is = %.4f',xf,y);
t=cputime-t;
fprintf('\n\nTotal Execution Time = %f\n\n',t);
```

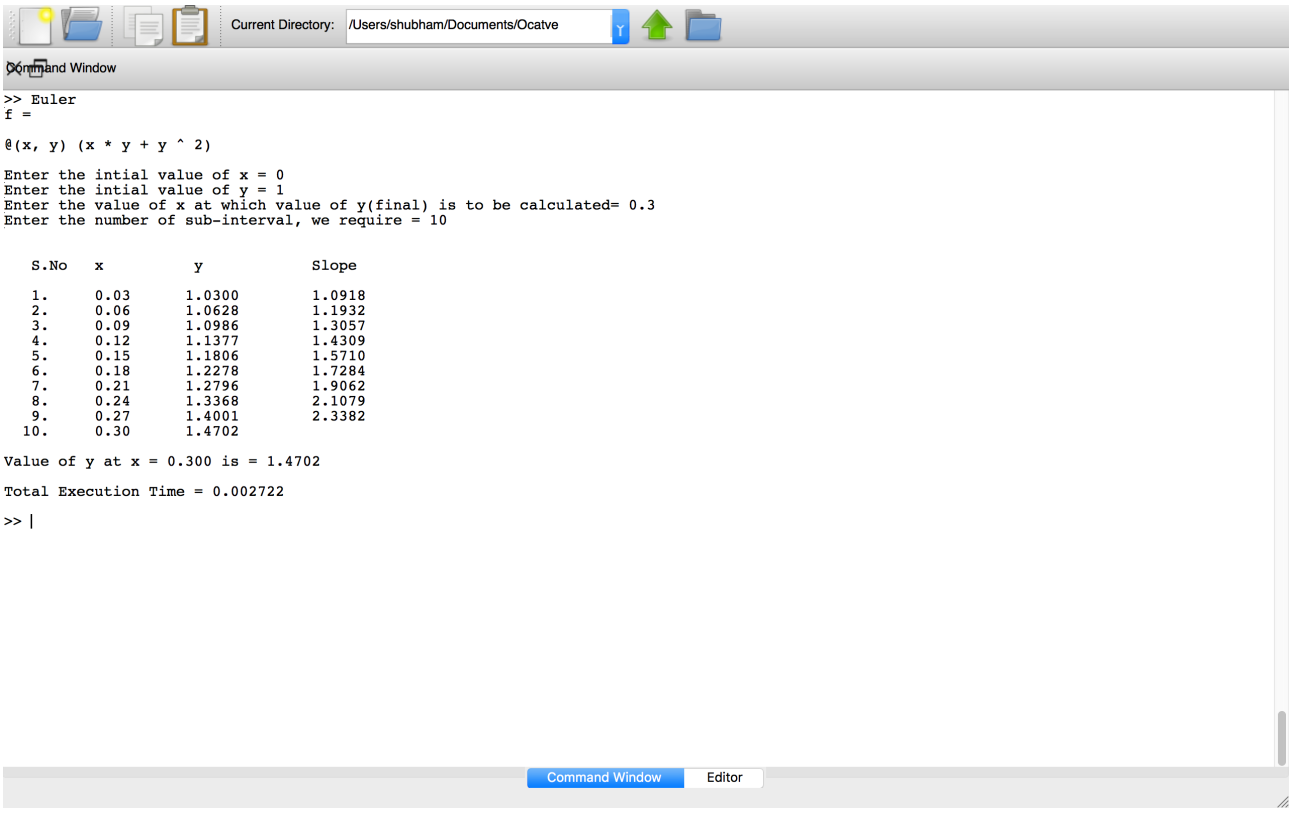
Outputs:

(a).



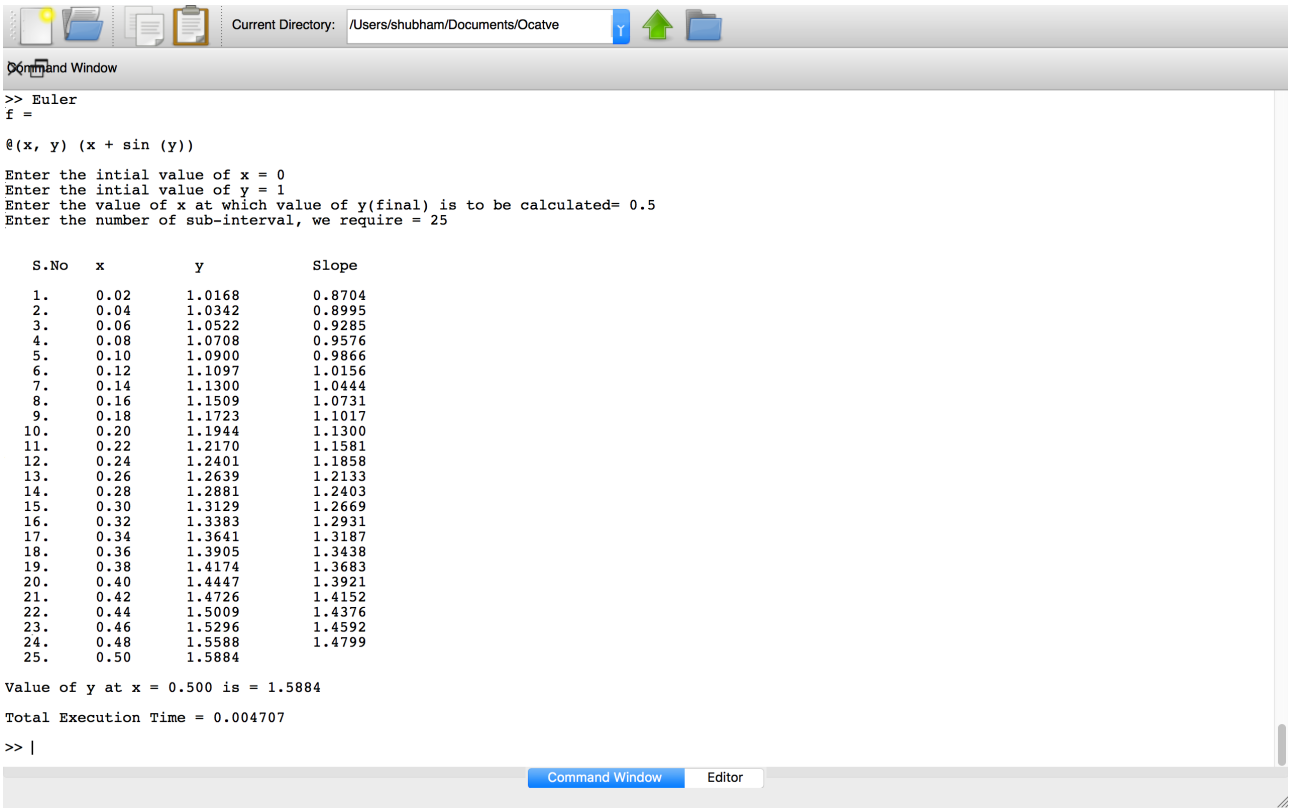

```
>> Euler
f =

@(x, y) (x + y)

Enter the intial value of x = 0
Enter the intial value of y = 1
Enter the value of x at which value of y(final) is to be calculated= 1
Enter the number of sub-interval, we require = 10


   S.No    x          y            Slope

    1.     0.10      1.1000        1.2000
    2.     0.20      1.2200        1.4200
    3.     0.30      1.3620        1.6620
    4.     0.40      1.5282        1.9282
    5.     0.50      1.7210        2.2210
    6.     0.60      1.9431        2.5431
    7.     0.70      2.1974        2.8974
    8.     0.80      2.4872        3.2872
    9.     0.90      2.8159        3.7159
   10.     1.00      3.1875

Value of y at x = 1.000 is = 3.1875

Total Execution Time = 0.002751

>>
```


(b).




```
>> Euler
f =

@(x, y) (x * y + y ^ 2)

Enter the intial value of x = 0
Enter the intial value of y = 1
Enter the value of x at which value of y(final) is to be calculated= 0.3
Enter the number of sub-interval, we require = 10


   S.No    x          y            Slope

    1.     0.03      1.0300        1.0918
    2.     0.06      1.0628        1.1932
    3.     0.09      1.0986        1.3057
    4.     0.12      1.1377        1.4309
    5.     0.15      1.1806        1.5710
    6.     0.18      1.2278        1.7284
    7.     0.21      1.2796        1.9062
    8.     0.24      1.3368        2.1079
    9.     0.27      1.4001        2.3382
   10.     0.30      1.4702

Value of y at x = 0.300 is = 1.4702

Total Execution Time = 0.002722

>>
```

(c).

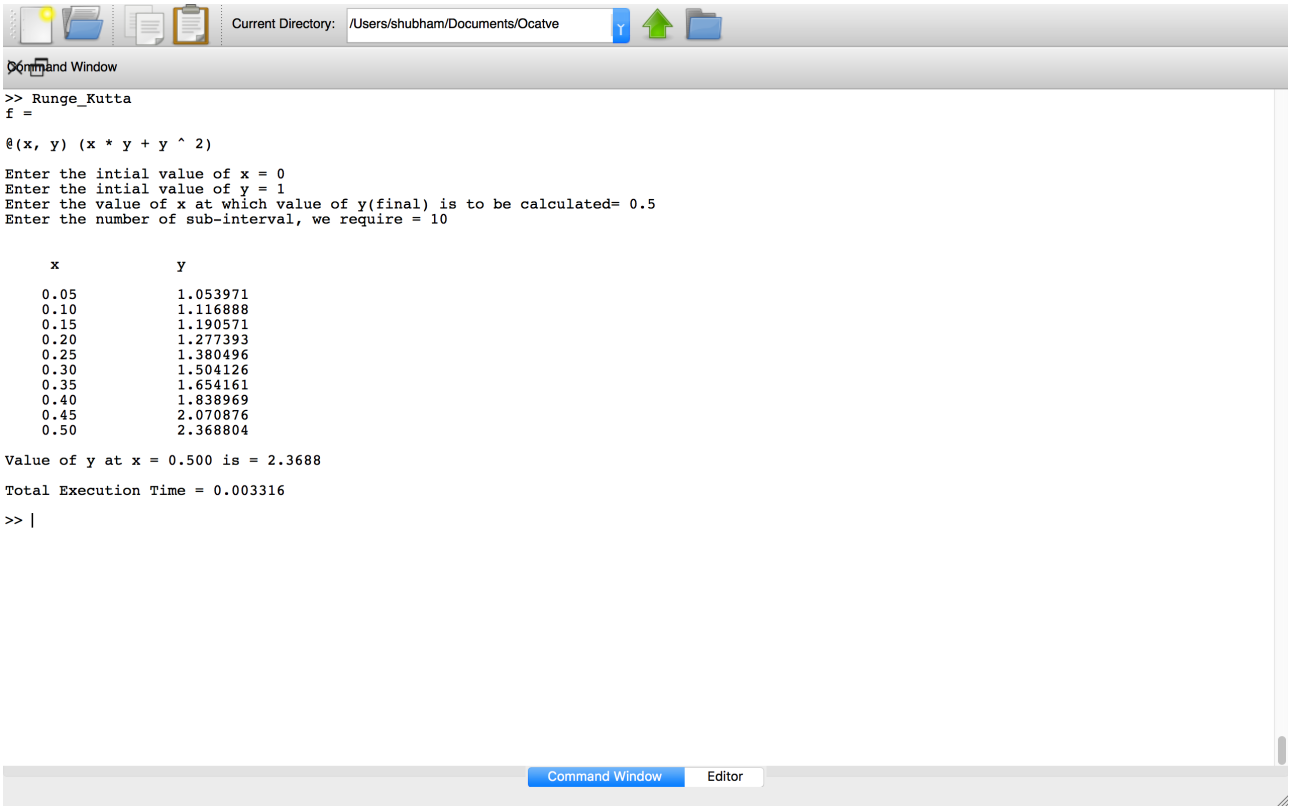

Current Directory:
/Users/shubham/Documents/Ocatve
Command Window
>> Euler
f =
@(x, y) (x + sin (y))
Enter the intial value of x = 0
Enter the intial value of y = 1
Enter the value of x at which value of y(final) is to be calculated= 0.5
Enter the number of sub-interval, we require = 25
S.No x y Slope
1. 0.02 1.0168 0.8704
2. 0.04 1.0342 0.8995
3. 0.06 1.0522 0.9285
4. 0.08 1.0708 0.9576
5. 0.10 1.0900 0.9866
6. 0.12 1.1097 1.0156
7. 0.14 1.1300 1.0444
8. 0.16 1.1509 1.0731
9. 0.18 1.1723 1.1017
10. 0.20 1.1944 1.1300
11. 0.22 1.2170 1.1581
12. 0.24 1.2401 1.1858
13. 0.26 1.2639 1.2133
14. 0.28 1.2881 1.2403
15. 0.30 1.3129 1.2669
16. 0.32 1.3383 1.2931
17. 0.34 1.3641 1.3187
18. 0.36 1.3905 1.3438
19. 0.38 1.4174 1.3683
20. 0.40 1.4447 1.3921
21. 0.42 1.4726 1.4152
22. 0.44 1.5009 1.4376
23. 0.46 1.5296 1.4592
24. 0.48 1.5588 1.4799
25. 0.50 1.5884
Value of y at x = 0.500 is = 1.5884
Total Execution Time = 0.004707
>>
Command Window
Editor

Write a Program to implement Runge-Kutta's Method

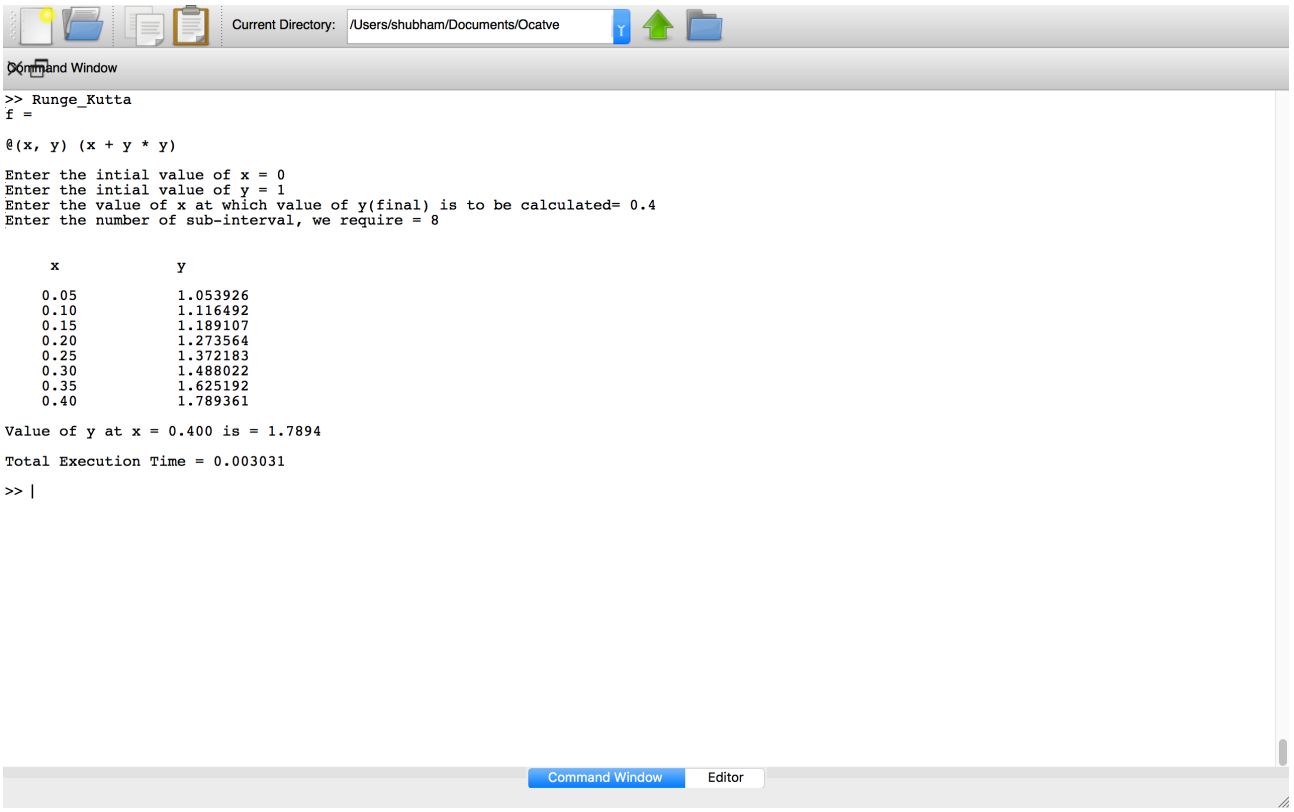
```
#=========================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to obtain solution of an equation using Runge-Kutta's Method.

#=========================================================================

f =@(x,y)(x*y+y^2)

  x0=input('Enter the intial value of x = ');

  y0=input('Enter the intial value of y = ');

  xf=input('Enter the value of x at which value of y(final) is to be calculated= ');

  n=input('Enter the number of sub-interval, we require = ');

  t=cputime;


# Runge-Kutta's Method Code

  x=x0;

  y=y0;

  h=(xf-x0)/n;

  fprintf('\n\n     x             y     \n\n');

  for i=1:n

      k1=h*f(x,y);

      k2=h*f(x+0.5*h,y+0.5*k1);

      k3=h*f(x+0.5*h,y+0.5*k2);

      k4=h*f(x+h,y+k3);

      k=(1/6)*(k1+2*k2+2*k3+k4);

      y=y+k;

      x=x+h;
```

```
    fprintf('    %.2f            %f\n',x,y);
end

fprintf('\nValue of y at x = %.3f is = %.4f',xf,y);
t=cputime-t;
fprintf('\n\nTotal Execution Time = %f\n\n',t);
```

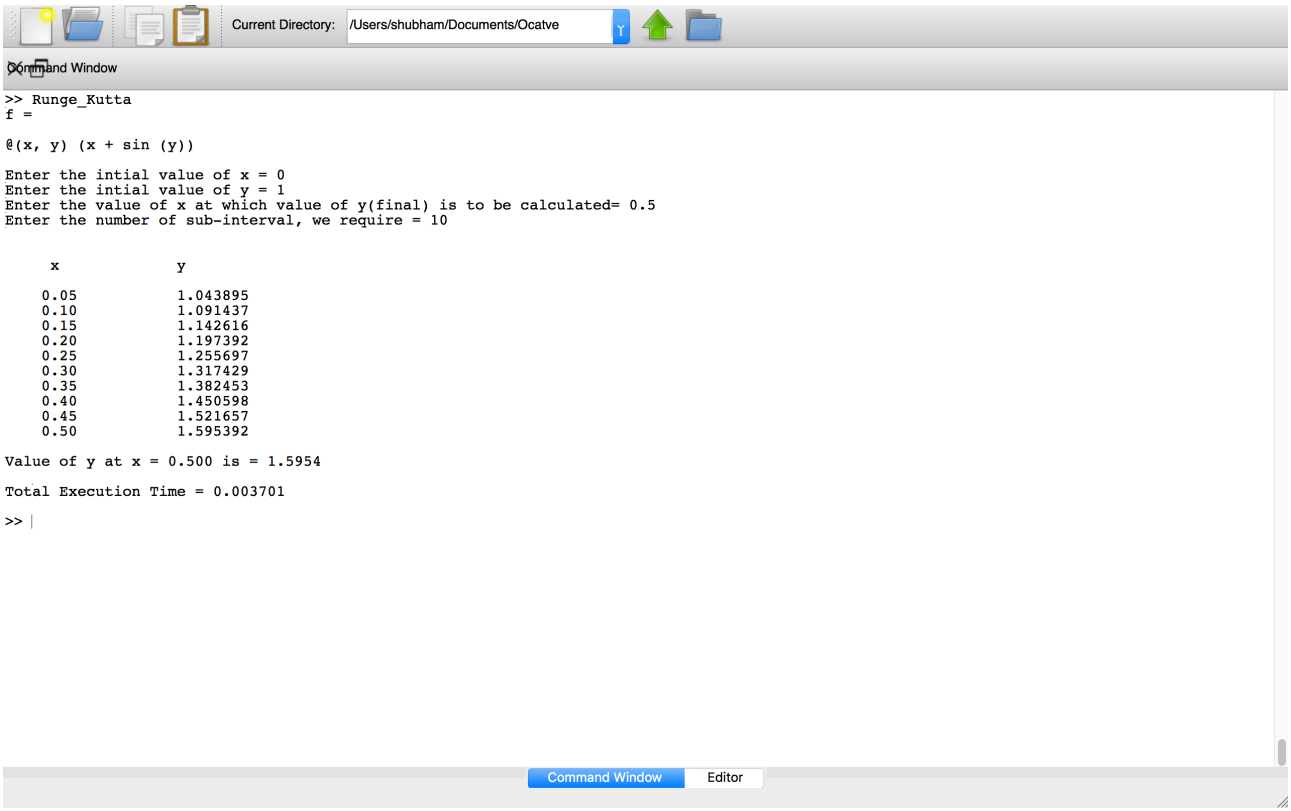
Outputs:
(a).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Runge_Kutta
f =
@(x, y) (x * y + y ^ 2)
Enter the intial value of x = 0
Enter the intial value of y = 1
Enter the value of x at which value of y(final) is to be calculated= 0.5
Enter the number of sub-interval, we require = 10
x y
0.05 1.053971
0.10 1.116888
0.15 1.190571
0.20 1.277393
0.25 1.380496
0.30 1.504126
0.35 1.654161
0.40 1.838969
0.45 2.070876
0.50 2.368804
Value of y at x = 0.500 is = 2.3688
Total Execution Time = 0.003316
>>
Command Window
Editor

(b).


Current Directory: /Users/shubham/Documents/Ocatve
Command Window
>> Runge_Kutta
f =
@(x, y) (x + y * y)
Enter the intial value of x = 0
Enter the intial value of y = 1
Enter the value of x at which value of y(final) is to be calculated= 0.4
Enter the number of sub-interval, we require = 8
x y
0.05 1.053926
0.10 1.116492
0.15 1.189107
0.20 1.273564
0.25 1.372183
0.30 1.488022
0.35 1.625192
0.40 1.789361
Value of y at x = 0.400 is = 1.7894
Total Execution Time = 0.003031
>>
Command Window
Editor

(c).

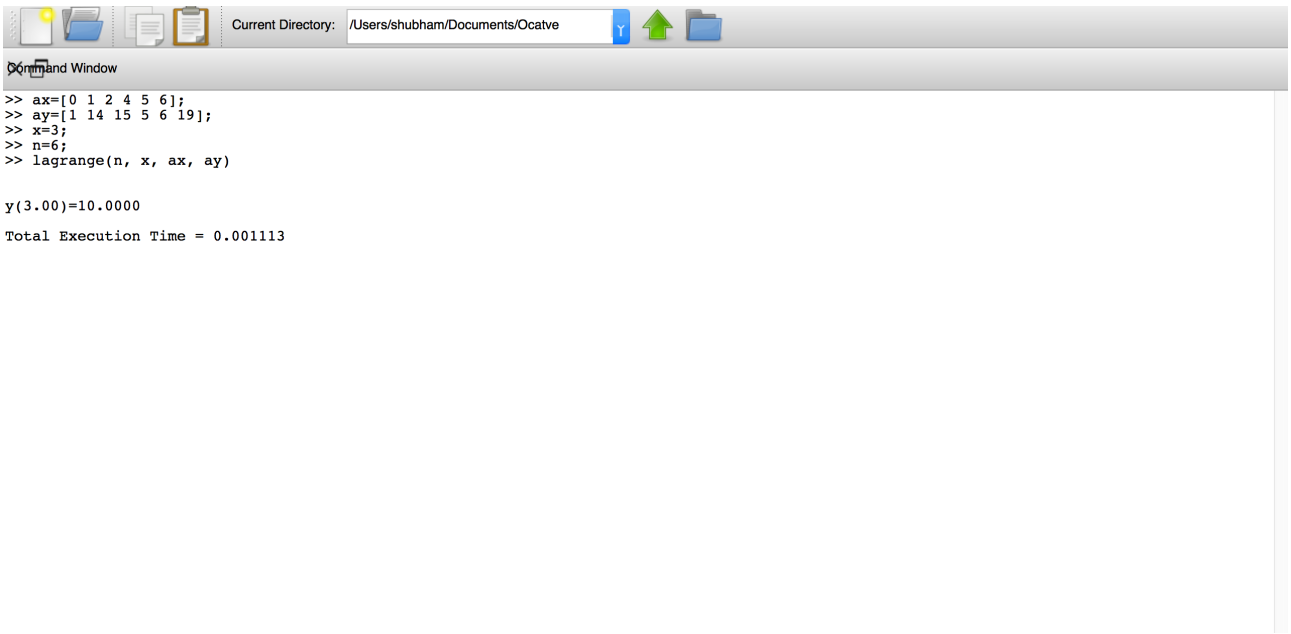

Current Directory:
/Users/shubham/Documents/Ocatve
Command Window
>> Runge_Kutta
f =
@(x, y) (x + sin (y))
Enter the intial value of x = 0
Enter the intial value of y = 1
Enter the value of x at which value of y(final) is to be calculated= 0.5
Enter the number of sub-interval, we require = 10
x y
0.05 1.043895
0.10 1.091437
0.15 1.142616
0.20 1.197392
0.25 1.255697
0.30 1.317429
0.35 1.382453
0.40 1.450598
0.45 1.521657
0.50 1.595392
Value of y at x = 0.500 is = 1.5954
Total Execution Time = 0.003701
>>
Command Window
Editor

Write a Program to implement Lagrange's Interpolation Method.

```
#=====================================================================

# Shubham Chahal

# 140020203052

# Numerical Methods - Lab

# Program to obtain solution of an equation using Lagrange's Interpolation Method.

#=====================================================================


function y=lagrange(n, x, ax, ay)

    # ax = Vector containing inputs (x values)
    # ay = Vector containing outputs (results for these x values)
    # x  = Value you want to compute, for interpolation
    # y  = Computed value
    # n  = No. of inputs

    t=cputime;
    y = 0;


    for i=1:n
        c = 1;
        d = 1;
        for j=1:n

            if (j != i)


              c=c*(x-ax(j));
              d=d*(ax(i)-ax(j));
            end


        end

        y = y + ay(i) * (c/d);
    end

    fprintf("\n\ny(%.2f)=%.4f",x,y);

    t=cputime-t;

    fprintf('\n\nTotal Execution Time = %f \n\n',t);
endfunction
```

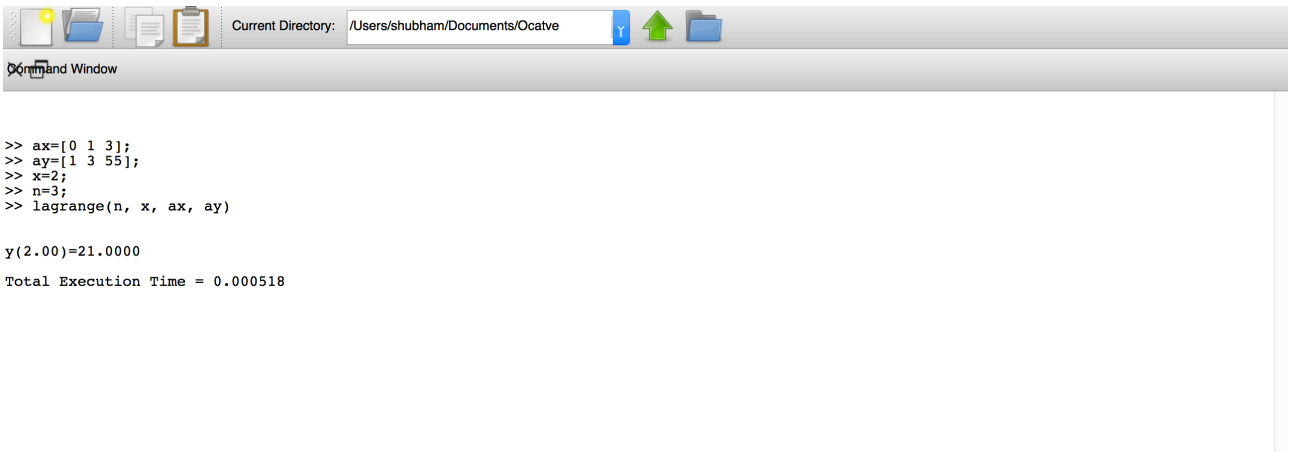
Outputs:
(a). Find f(2) for the data:

x	0	1	3
f(x)	1	3	55



(b). Find f(3) for the data:

x	0	1	2	4	5	6
f(x)	1	14	15	5	6	19



(c). Find f(0.25) for the data:

x	0.1	0.2	0.3	0.4	0.5
f(x)	9.9833	4.9667	3.2836	2.4339	1.9177