APPLIED MATHEMATICS LAB

ETCS-252

Faculty Name : Mr. Braham Prakash

Student Name : Khushi
Roll No : 03114813120
Branch : ITE
Group : 2
Semester : 4th

PRACTICAL RECORD

Exp No.	Experiment Name	Date of Performance	Marks/ Remarks
1(a).	Write Scilab to perform addition of two matrices.		
1(b).	Program to perform multiplication of two matrices.		
1(c).	Program to find transpose of a matrix.		
2.	Program to find inverse of a matrix using Gauss Jordan Method		
3.	Program to find Eigen Values and Eigen Vectors of a given 2x2 matrix.		
4.	Program to find mean, standard deviation and first r moments about mean of given grouped data.		
5(a).	To fit a straight line for given data.		
5(b).	To fit a parabola from a given data.		

6(a).	Program to plot unit step function.			
6(b).	Program to plot a square wave function,			
7(a).	Program to find solution of a non- linear equation using Bisection method.			
7(b).	Program to find solution of a non-linear equation using Newton - Raphson method.			
8(a).	Program to evaluate a definite integral using Trapezoidal rule.			
8(b).	Program to evaluate a definite integral using Simpson's one third rule.			
8(c).	Program to evaluate a definite integral using Simpson's three -eigth rule.			
9.	Program to find the initial value problem using Euler's method.			
10.	Program to find the solution of initial value using Runge-Kutta method of fourth order.			

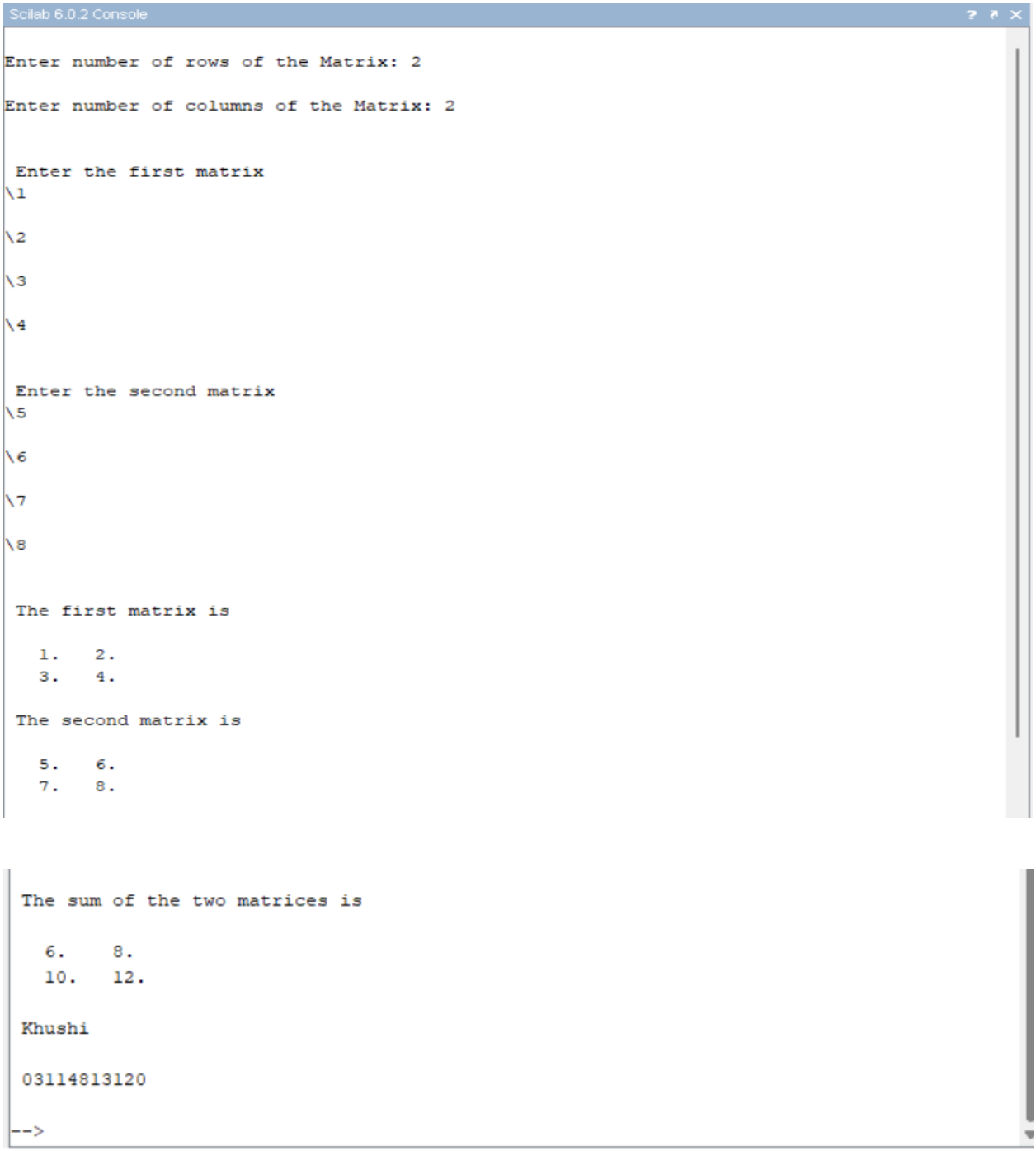
EXPERIMENT - 1

a) AIM: Addition of two matrices.

Scilab code:

```
//Matrix Addition
clc
m=input("Enter number of rows of the Matrix: ");
n=input("Enter number of columns of the Matrix: ");
disp('Enter the first matrix')
for i=1:m
    for j=1:n
        A(i,j)=input('\');
        end
end

disp('Enter the second matrix')
for i=1:m
    for j=1:n
        B(i,j)=input('\');
        end
end

for i=1:m
    for j=1:n
        C(i,j)=A(i,j)+B(i,j);
        end
end

disp('The first matrix is')
disp(A)
disp('The second matrix is')
disp(B)
disp('The sum of the two matrices is')
disp(C)

disp('Khushi')
disp('03114813120')
```

OUTPUT:

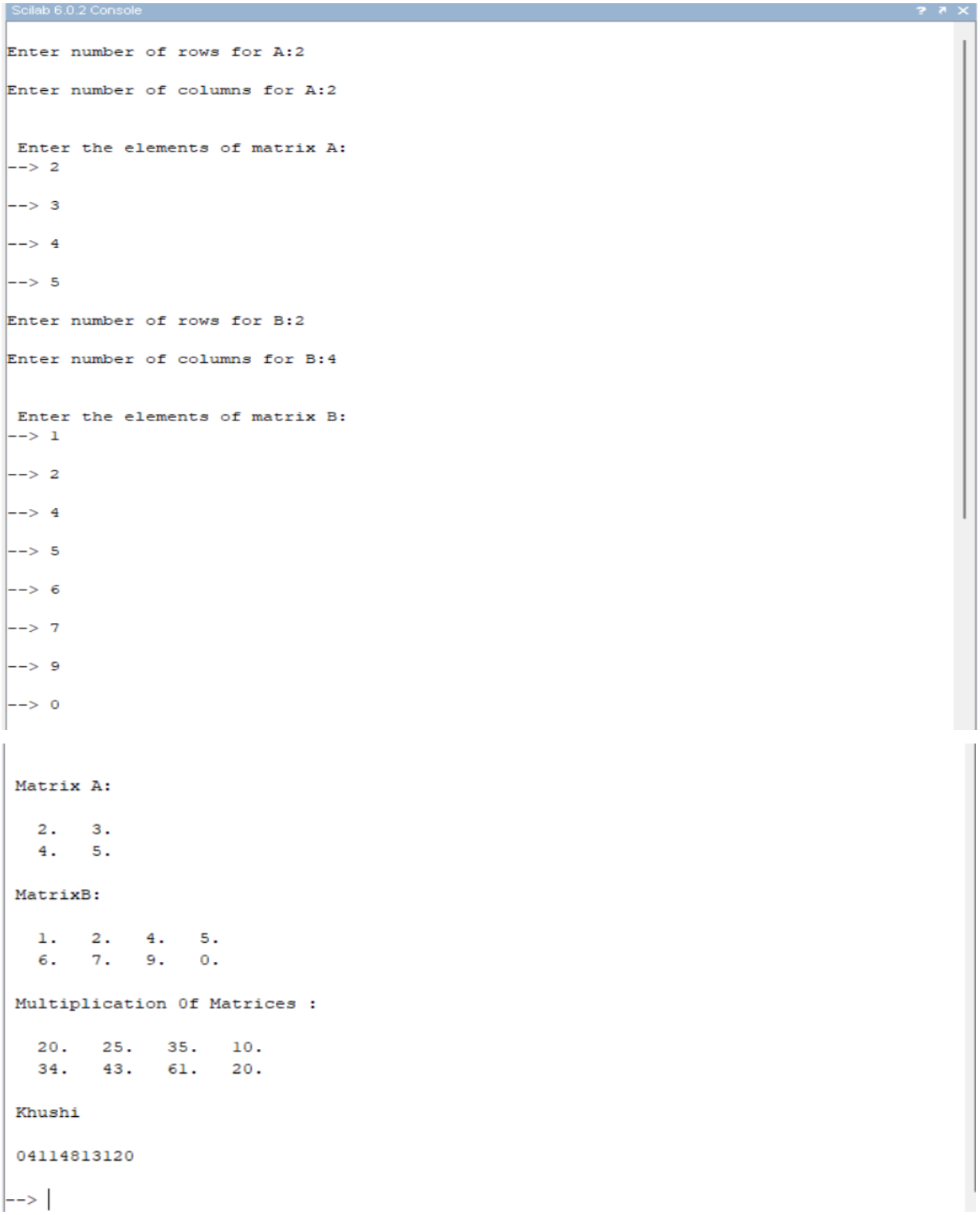

Scilab 6.0.2 Console
Enter number of rows of the Matrix: 2
Enter number of columns of the Matrix: 2
Enter the first matrix
\1
\2
\3
\4
Enter the second matrix
\5
\6
\7
\8
The first matrix is
1. 2.
3. 4.
The second matrix is
5. 6.
7. 8.
The sum of the two matrices is
6. 8.
10. 12.
Khushi
03114813120
-->

b) AIM : Multiplication of two Matrices

Scilab Code:

```
//Matrix multiplication
clc
m=input("Enter the number of rows of first matrix :");
n=input("Enter the number of columns of first matrix :");
p=input("Enter the number of rows of second matrix :");
q=input("Enter the number of columns of second matrix :");

if n==p then
   disp('Matrices are conformable for Multiplication.')
else
   disp('Matrices are not conformable for Multiplication.')
   break;
end

disp('Enter the first matrix :')
for i=1:m
   for j=i:n
      A(i,j)=input('\');
      end
end

disp('Enter the second matrix :')
for i=1:p
   for j=i:q
      B(i,j)=input('\');
      end
end

C=zeroes(m,q);
for i=1:m
   for j=i:q
      for k =1:n
      C(i,j)=C(i,j)+A(i,k)*B(k,j);
   end
   end
end

disp('The first matrix is :')
disp(A)
disp('The second matrix is :')
disp(B)
disp('The product of two matrices is :')
disp(C)

disp('Khushi ')
disp('03114813120')
```

OUTPUT :

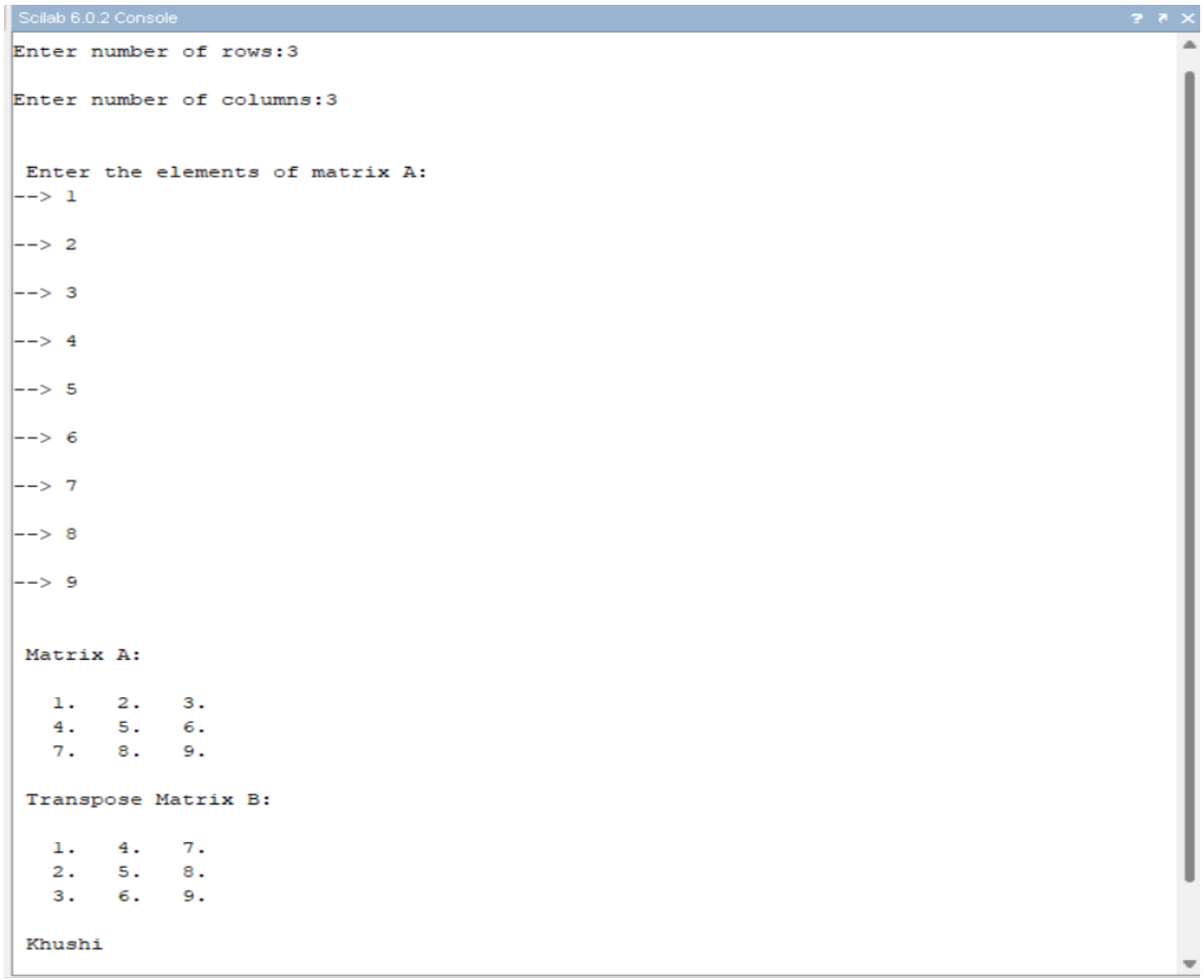

Scilab 6.0.2 Console
Enter number of rows for A:2
Enter number of columns for A:2
Enter the elements of matrix A:
--> 2
--> 3
--> 4
--> 5
Enter number of rows for B:2
Enter number of columns for B:4
Enter the elements of matrix B:
--> 1
--> 2
--> 4
--> 5
--> 6
--> 7
--> 9
--> 0
Matrix A:
2. 3.
4. 5.
MatrixB:
1. 2. 4. 5.
6. 7. 9. 0.
Multiplication 0f Matrices :
20. 25. 35. 10.
34. 43. 61. 20.
Khushi
04114813120
-->

c) AIM : Transpose of a Matrix

Scilab Code:

```
//Transpose Of A Matrix
clc

m=input("Enter number of rows:")
n=input("Enter number of columns:")
a=zeros(m,n)
b=zeros(n,m)

disp("Enter the elements of matrix A:");
for i=1:m
for j=1:n
   a(i,j)=input("");
end;
end;

for i=1:m
   for j=1:n
       b(j,i)=a(i,j);
    end;
    end;

disp("Matrix A:");
disp(a);
disp("Transpose Matrix B:");
disp(b);

disp('Khushi')
disp('03114813120')
```

OUTPUT:


Scilab 6.0.2 Console
Enter number of rows:3
Enter number of columns:3
Enter the elements of matrix A:
--> 1
--> 2
--> 3
--> 4
--> 5
--> 6
--> 7
--> 8
--> 9
Matrix A:
1. 2. 3.
4. 5. 6.
7. 8. 9.
Transpose Matrix B:
1. 4. 7.
2. 5. 8.
3. 6. 9.
Khushi

EXPERIMENT - 2

AIM : Scilab code for inverse of a matrix using Gauss Jordan Method.

Scilab Code:

```
// Inverse of a matrix using Gauss-Jordan.
 clc
 disp('Enter a 3 by 3 matrix row-wise, make sure that diagonal
 elements are non -zeros')
 for i=1:3
 for j=1:3
 A(i,j)=input('\');
  end
 end
 disp('Entered
 Matrix is')
 disp(A)
 if det(A)==0
 disp('Matrix is singular, Inverse does not
 exist') break;
 end
 //Taking the augmented
 matrix [A|I], B=[A eye(3,3)]
 disp('Augumented
 matrix is:') disp(B)
 //Making
 B(1,1)=1 B(1,:)
 = B(1,:)/B(1,1);
 //Making B(2,1) and
 B(3,1)=0 B(2,:) = B(2,:)
 - B(2,1)*B(1,:);
 B(3,:) = B(3,:) - B(3,1)*B(1,:);
 //Making B(2,2)=1 and B(1,2),
 B(3,2)=0 7
```

```
B(2,:) = B(2,:)/B(2,2);
B(1,:) = B(1,:) - B(1,2)*B(2,:);
B(3,:) = B(3,:) - B(3,2)*B(2,:);
// Making B(3,3)=1 and B(1,3),
B(2,3)=0 B(3,:) = B(3,:)/B(3,3);
B(1,:) = B(1,:) - B(1,3)*B(3,:);
B(2,:) = B(2,:) - B(2,3)*B(3,:);
disp('Augumented matrix after row operations is:')
disp(B)
B(:,1:3)=[]
disp(‘Inverse of the Matrix is’)
disp(B)
disp(‘Khushi’)
disp(‘03114813120’)
```

OUTPUT:

```
"Enter a 3 by 3 matrix row-wise, make sure that diagonal elements are non -zeros"
\1

\2

\3

\4

\5

\6

\1

\3

\4


"Entered Matrix is"

 1.   2.   3.
 4.   5.   6.
 1.   3.   4.

"Augumented matrix is:"

 1.   2.   3.   1.   0.   0.
 4.   5.   6.   0.   1.   0.
 1.   3.   4.   0.   0.   1.

"Augumented matrix after row operations is:"
```

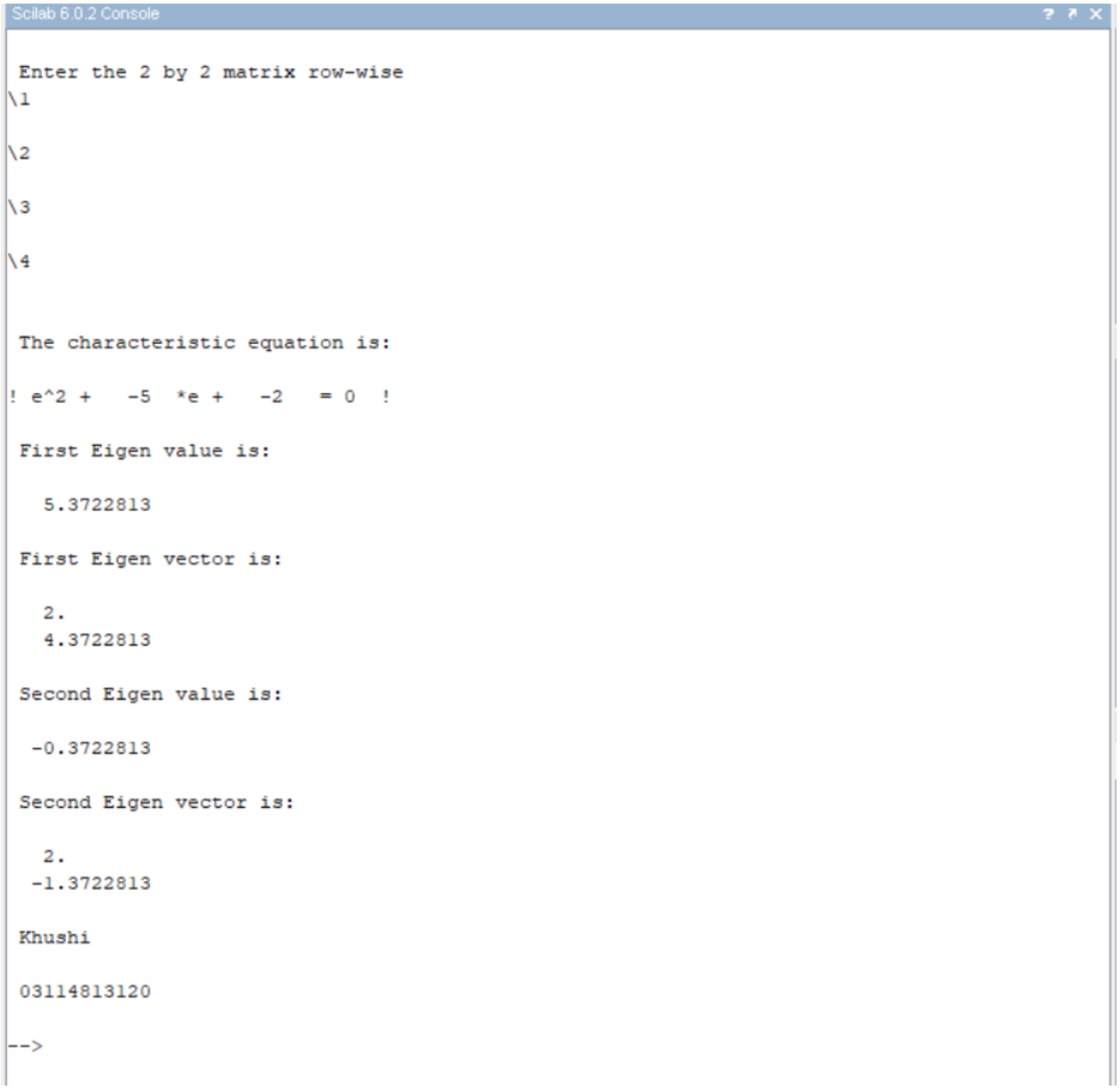
EXPERIMENT - 3

AIM : Scilab Code to find Eigen values of a 2 X 2 matrix.

Scilab Code:

```
//Eigen values of a  2*2 matrix
clc
disp('Enter the 2 by 2 matrix row-wise')
for i=1:2
   for j=1:2
      A(i,j)=input('\');
      end
end
b=A(1,1)+A(2,2);
c=A(1,1)*A(2,2)-A(1,2)*A(2,1);
disp('The characteristic equation is:')
disp([' e^2 + ' string(-b) '*e + ' string(c) ' = 0'])
e1=(b+sqrt(b^2-4*c))/2;
e2=(b-sqrt(b^2-4*c))/2;
if A(1,2) ~= 0
 v1 = [A(1,2); e1-A(1,1)];
 v2 = [A(1,2); e2-A(1,1)];
 elseif A(2,1) ~= 0
 v1 = [e1-A(2,2); A(2,1)];
 v2 = [e2-A(2,2); A(2,1)];
 else
 v1 = [1; 0];
 v2 = [0; 1];
end
 disp('First Eigen value is:');
 disp(e1)
 disp('First Eigen vector is:');
 disp (v1)
 disp('Second Eigen value is:');
 disp(e2)
 disp('Second Eigen vector is:');
 disp(v2)
 disp('Khushi')
 disp('03114813120')
```

OUTPUT:


Scilab 6.0.2 Console
Enter the 2 by 2 matrix row-wise
\1
\2
\3
\4
The characteristic equation is:
! e^2 + -5 *e + -2 = 0 !
First Eigen value is:
5.3722813
First Eigen vector is:
2.
4.3722813
Second Eigen value is:
-0.3722813
Second Eigen vector is:
2.
-1.3722813
Khushi
03114813120
-->

EXPERIMENT - 4

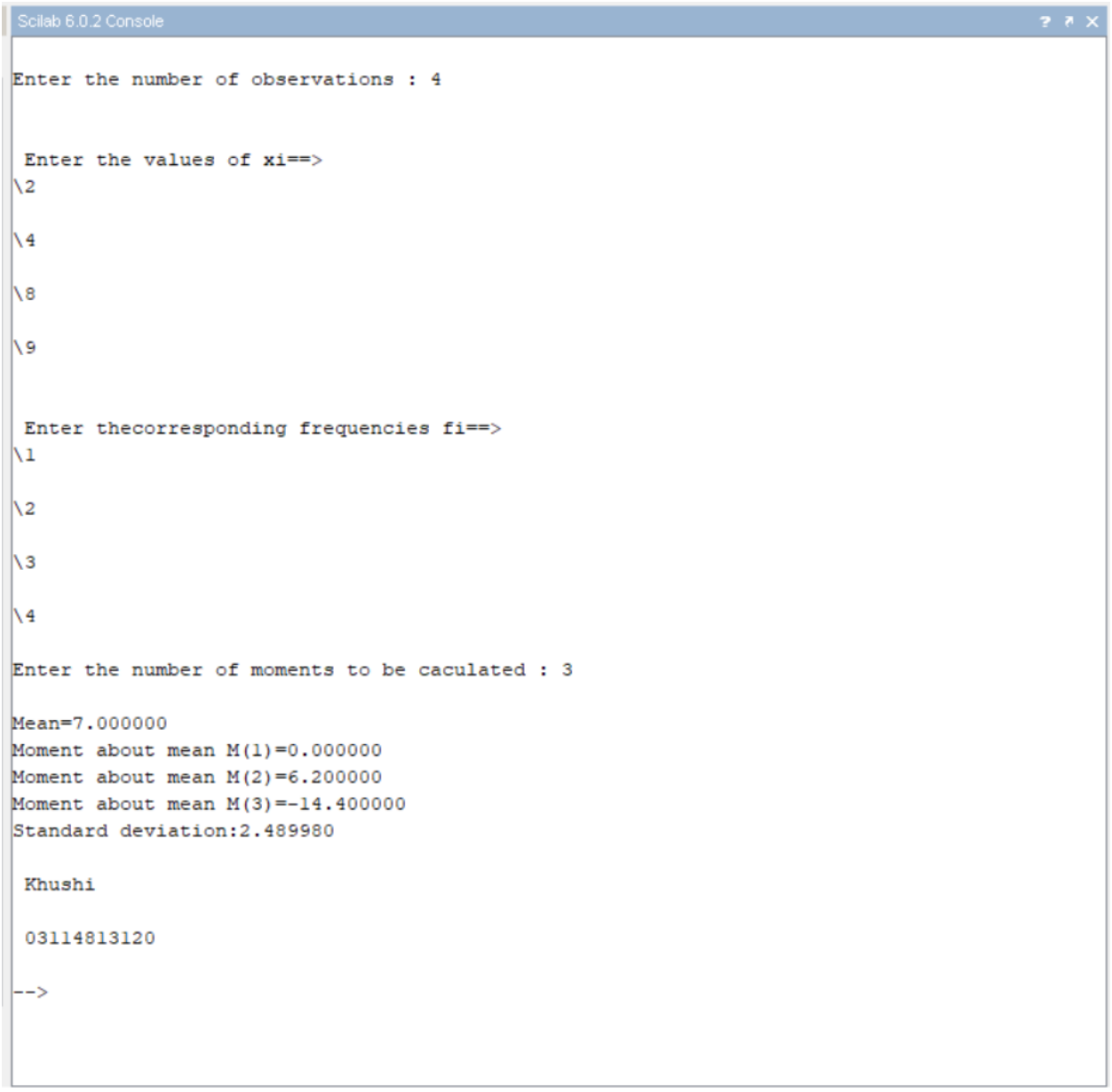
AIM : Scilab code to find Mean, Standard Deviation and Moment about Mean of a frequency data.

Scilab code:

```
//Mean, Standard deviation and Moment about Mean
clc;
n=input('Enter the number of observations : ');
disp('Enter the values of xi==>');

for i=1:n
   x(i)=input('\');
   end;

disp('Enter the corresponding frequencies fi==>');
sum3=0;
for i=1:n
   f(i)=input('\');
   sum3=sum3+f(i);
   end;

r=input('Enter the number of moments to be calculated : ');
sum1=0;

for i=1:n
   sum1=sum1+f(i)*x(i);
   end;

A=sum1/sum3;
printf('Mean=%f\n',A);  //calculation of mean

for j=1:r
   sum2=0;

for i=1:n
   y(i)=f(i)*(x(i)-A)^j;
   sum2=sum2+y(i);
end;

M(j)=(sum2/sum3); //calculation of moments
printf('Moment about mean M(%d)=%f\n',j,M(j));
end;
```

```
sd=sqrt(M(2)); //calculation of standard deviation
printf('Standard deviation:%f\n',sd)

disp('Khushi')
disp('03114813120')
```

OUTPUT:



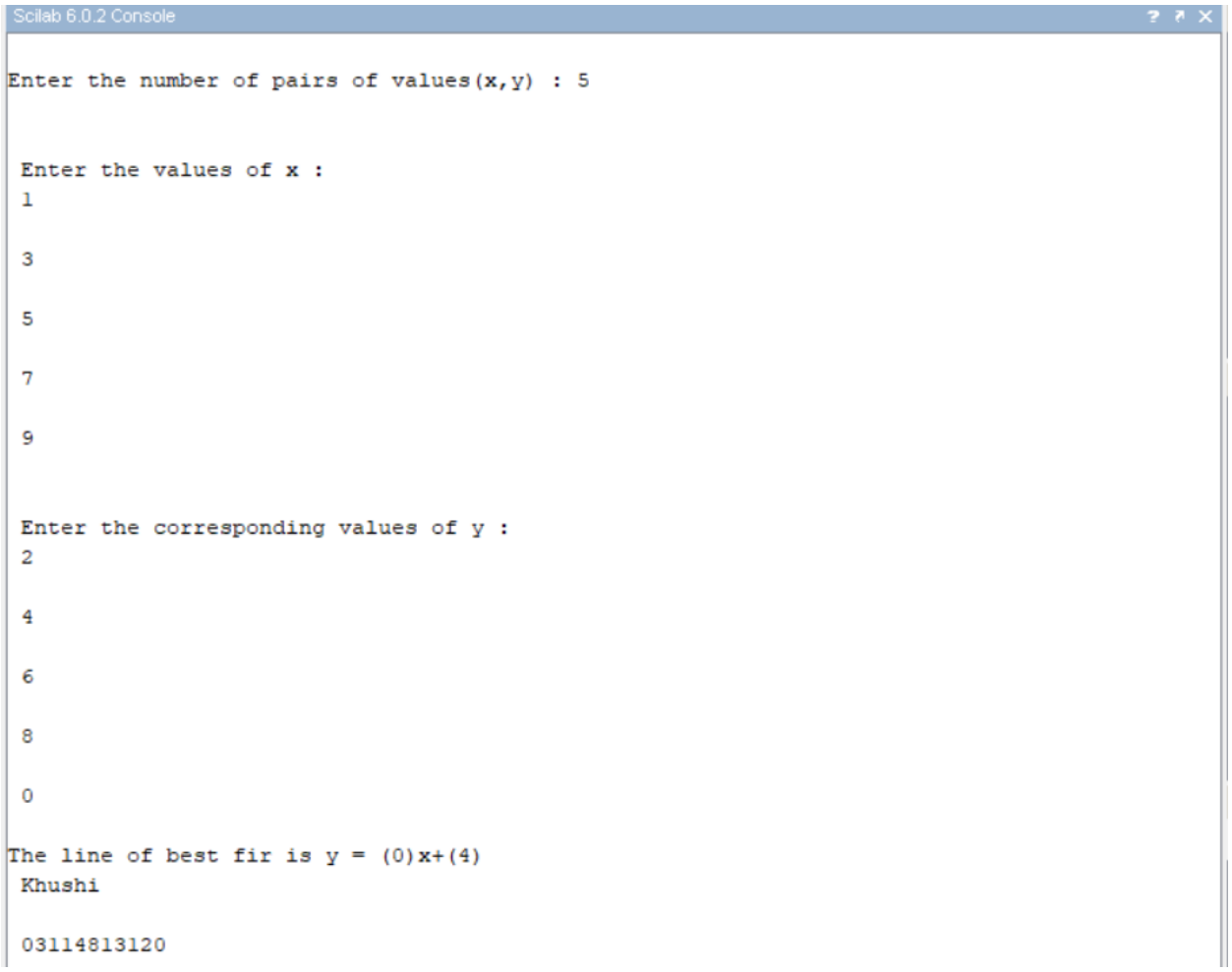
EXPERIMENT - 5(a)

AIM: Scilab code to fit a straight line from given data.

Scilab Code:

```
//Fit a straight line from given data
clc
n=input('Enter the number of pairs of values(x,y) : ')
disp('Enter the values of x :  ')

for i=1:n
   x(i)=input(' ')
end

disp('Enter the corresponding values of y : ')
for i=1:n
   y(i)=input(' ')
end

sumx=0; sumx2=0; sumy=0; sumxy=0;

for i=1:n
  sumx=sumx+x(i);
  sumx2=sumx2+x(i)*x(i);
  sumy=sumy+y(i);
  sumxy=sumxy+x(i)*y(i);
end

A=[sumx n; sumx2 sumx];
B=[sumy; sumxy];
C=inv(A)*B

printf('The line of best fir is y = (%g)x+(%g)',C(1,1),C(2,1))

disp('Khushi')
disp('03114813120')
```

OUTPUT:


Scilab 6.0.2 Console
Enter the number of pairs of values(x,y) : 5
Enter the values of x :
1
3
5
7
9
Enter the corresponding values of y :
2
4
6
8
0
The line of best fir is y = (0)x+(4)
Khushi
03114813120

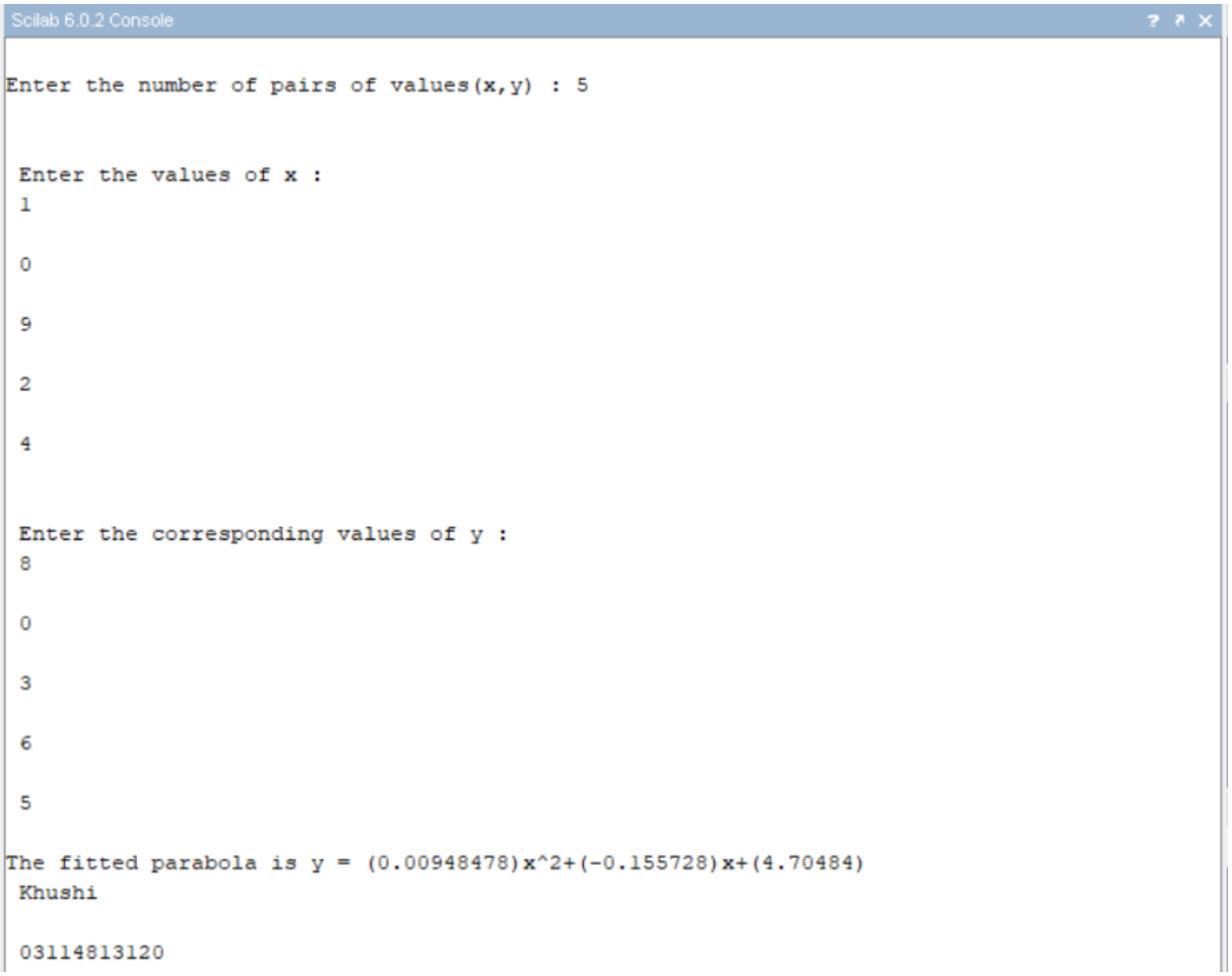
EXPERIMENT - 5(b)

AIM: Scilab code to fit a Parabola from given data.

Scilab Code:

```
//Fit a parabola from given data
clc
n=input('Enter the number of pairs of values(x,y) : ')
disp('Enter the values of x :')

for i=1:n
    x(i)=input(' ')
end

disp('Enter the corresponding values of y :')
for i=1:n
    y(i)=input(' ')
end

sumx=0; sumx2=0; sumx3=0; sumx4=0; sumy=0; sumxy=0; sumx2y=0;

for i=1:n
  sumx=sumx+x(i);
  sumx2=sumx2+x(i)*x(i);
  sumx3=sumx3+x(i)*x(i)*x(i);
  sumy=sumy+y(i);
  sumxy=sumxy+x(i)*y(i);
  sumx2y=sumx2y+x(i)*x(i)*y(i);
end

A=[sumx2 sumx n; sumx3 sumx2 sumx; sumx4 sumx3 sumx2];
B=[sumy; sumxy; sumx2y];
C=inv(A)*B

printf('The fitted parabola is y = (%g)x^2+(%g)x+(%g)',C(1,1),C(2,1),C(3,1))

disp('Khushi')
disp('03114813120')
```

OUTPUT:


Scilab 6.0.2 Console
Enter the number of pairs of values(x,y) : 5
Enter the values of x :
1
0
9
2
4
Enter the corresponding values of y :
8
0
3
6
5
The fitted parabola is y = (0.00948478)x^2+(-0.155728)x+(4.70484)
Khushi
03114813120

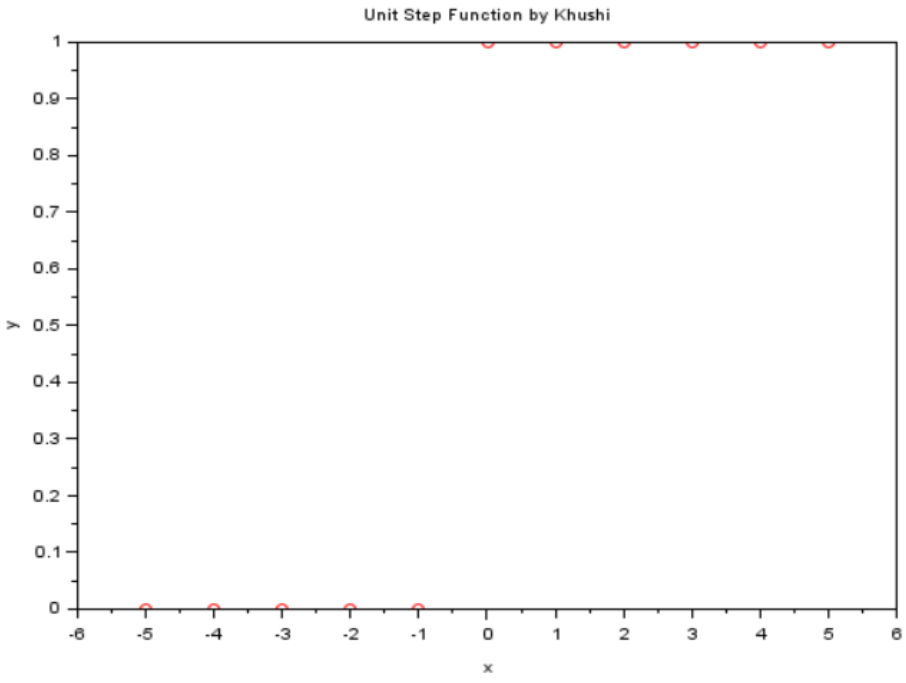
EXPERIMENT - 6(a)

AIM: Scilab code to plot Unit Step Function.

Scilab Code:

```
//Unit Step Function
clc
clc
x=[-1 -2 -3 -4 -5 0 1 2 3  4 5];
y=[0 0 0 0 0 1 1 1 1 1 1];
plot (x,y,'ro')
xlabel('x');
ylabel('y');
title('Unit Step Function by Khushi');

disp('Khushi')
disp('03114813120')
```

OUTPUT:



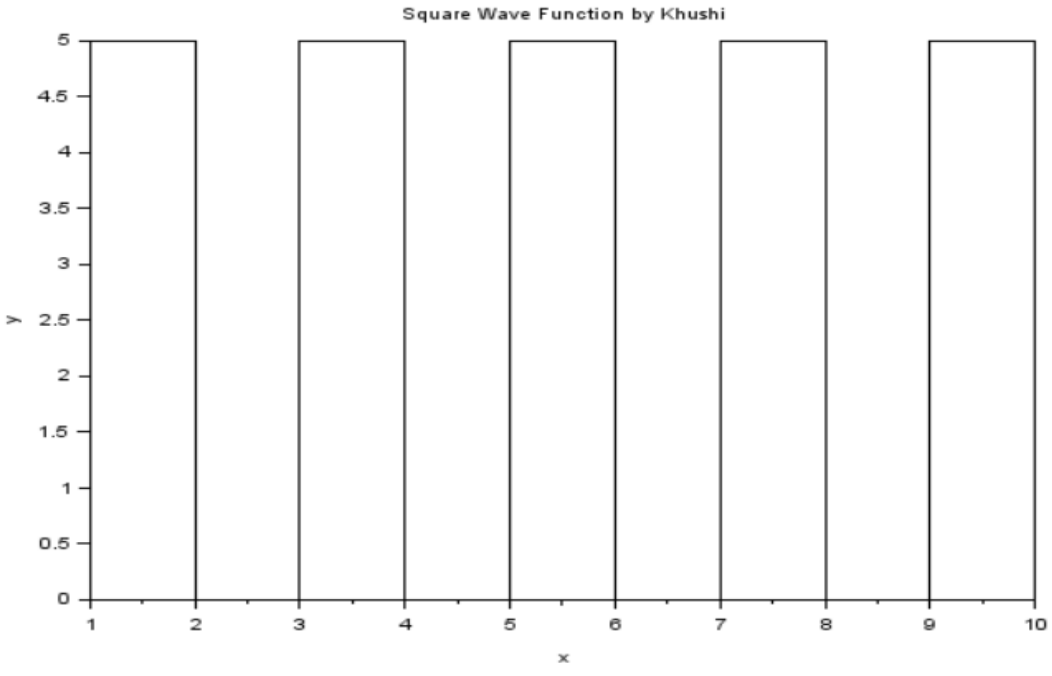
EXPERIMENT - 6(b)

AIM: Scilab code to plot Square Wave Function.

Scilab Code:

```
//Square Wave Function
clc
x = [1 2 3 4 5 6 7 8 9 10];
y = [5 0 5 0 5 0 5 0 5 0];
plot2d2(x,y);

xlabel('x');

ylabel('y');

title('Square Wave Function by Khushi ');
```

OUTPUT:



EXPERIMENT - 7(a)

AIM: Scilab code to solve non-linear equation using Bisection method.

Scilab Code:

```
//Bisection method
clc
deff('y=f(x)','y=x^3+x^2-3*x-3')
a=input("Enter initial interval value : ");
b=input("Enter final interval value : ");

fa = f(a); //compute initial values of f(a) and f(b)
fb = f(b);

if sign(fa) == sign(fb) // sanity check: f(a) and f(b) must have different signs
disp('f must have different signs at the endpoints a and b')
error
end

e=input(" answer correct upto : ");
iter=0;
printf('Iteration\ta\t\tb\t\troot\t\tf(root)\n')
while abs(a-b)>2*e

root=(a+b)/2

printf(' %i\t\t%f\t%f\t%f\t%f\n',iter,a,b,root,f(root))
if f(root)*f(a)>0

a=root
else b=root
end
iter=iter+1
end

printf('\n\nThe solution of given equation is %f after %i Iterations',root,iter-1)

disp('Khushi')
disp('03114813120')
```

OUTPUT:

Scilab 6.0.2 Console

```
Enter initial interval value : 1

Enter final interval value : 4

 answer correct upto : 0.001

Iteration       a               b               root            f(root)
 0              1.000000        4.000000        2.500000        11.375000
 1              1.000000        2.500000        1.750000        0.171875
 2              1.000000        1.750000        1.375000        -2.634766
 3              1.375000        1.750000        1.562500        -1.431396
 4              1.562500        1.750000        1.656250        -0.682220
 5              1.656250        1.750000        1.703125        -0.268597
 6              1.703125        1.750000        1.726563        -0.051755
 7              1.726563        1.750000        1.738281        0.059206
 8              1.726563        1.738281        1.732422        0.003513
 9              1.726563        1.732422        1.729492        -0.024174
 10             1.729492        1.732422        1.730957        -0.010344


The solution of given equation is 1.730957 after 10 Iterations
 Khushi

 03114813120
```

EXPERIMENT - 7(b)

AIM: Scilab code to solve non-linear equation using Newton Raphson method.

Scilab Code:

```
//Newton Raphson
clc;
deff('y=f(x)','y=x^3+x^2-3*x-3')
deff('y=df(x)','y=3*x^2+2*x-3')
x(1)=input('Enter Initial Guess:');
e= input("Answer correct upto : ");
for i = 1 : 100
x(i+1)=x(i)-f(x(i))/df(x(i));

err(i)=abs((x(i+1)-x(i))/x(i));
if err(i) < e
   break;
end
end

printf('The solution is %f',x(i))

disp('Khushi');
disp('03114813120');
```

OUTPUT:

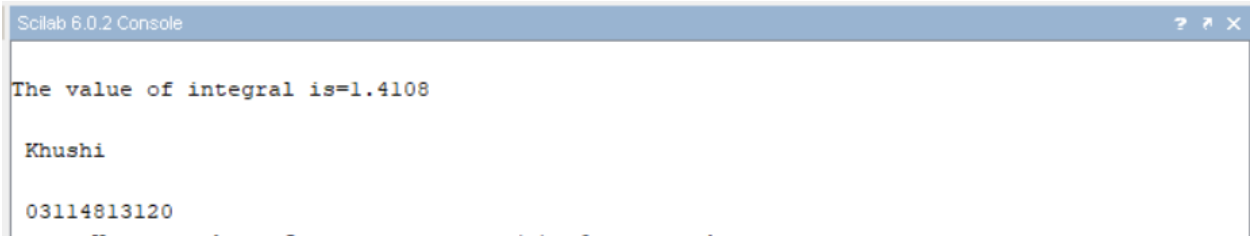
EXPERIMENT - 8(a)

AIM: Scilab code to find the value of definite integral using Trapezoidal method.

Scilab Code:

```
//Trapezoidal Rule
clc; clear; close;

deff('y=f(x)','y=1/(1+x^2)')
x0=0;
xn=6;
n=6;

h=(xn-x0)/n;
s=0;

for i=1:n
s=s+f(x0+(i-1)*h)+f(x0+i*h);
end

integral=(h*s)/2;

printf('\nThe value of integral is=%g\n',integral)

disp('Khushi');
disp('03114813120');
```

OUTPUT:



EXPERIMENT - 8(b)

AIM: Scilab code to find the value of definite integral using Simpsons 1/3rd Rule.

Scilab Code:

```
//Simphsons 1/3rd Rule
clc; clear; close;
deff('y=f(x)','y=sin(x)')
x0=0;
xn=%pi;
n=10;
//n should be even
h=(xn-x0)/n;
s=0;
for i=1:2:n
s=s+f(x0+(i-1)*h)+4*f(x0+i*h)+f(x0+(i+1)*h);
end
integral=(h*s)/3;

printf('\nThe value of integral is=%g\n',integral)

disp('Khushi');
disp('03114813120');
```

OUTPUT:

EXPERIMENT - 8(c)

AIM: Scilab code to find the value of definite integral using Simpsons 3/8 Rule.

Scilab Code:

```
//Simpsons 3/8 rule
clc; clear; close;
deff('y=f(x)','y=1/(1+x^2)')
 x0=0;
xn=6;
n=6;
//n should be a multiple of three
h=(xn-x0)/n;
s=0;
for i=1:3:n
s=s+f(x0+(i-1)*h)+3*f(x0+i*h)+3*f(x0+(i+1)*h)+f(x0+(i+2)*h);
end
integral=(3*h*s)/8;
printf('\nThe value of integral is=%g\n',integral)

disp('Khushi');
disp('03114813120');
```

OUTPUT:

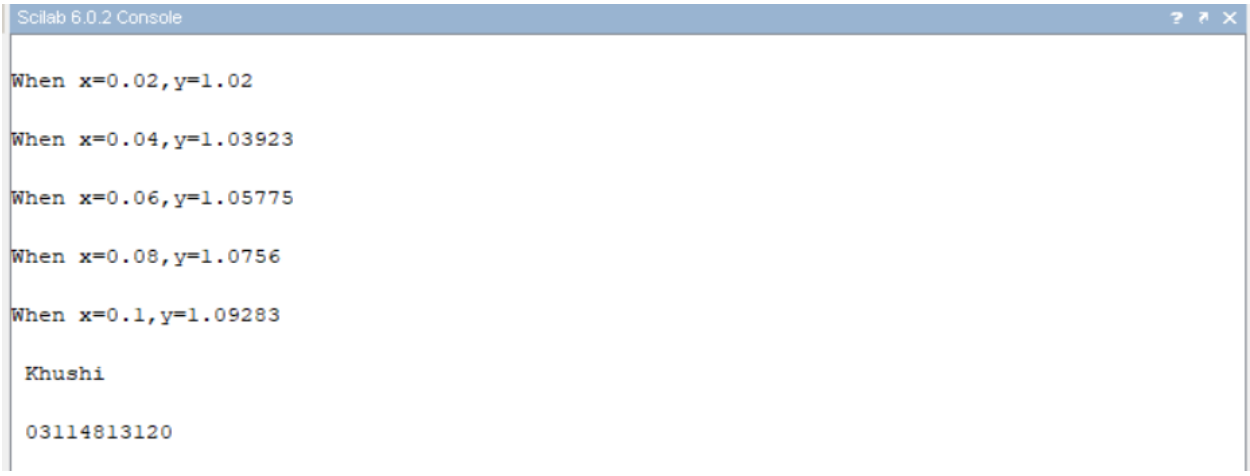
EXPERIMENT - 9

AIM: Scilab code to solve ordinary differential equations using Euler's method.

Scilab Code:

```
//Program to solve the equation y'=f(x,y); y(x0)=0 for y(xn) using Euler's method
clc;
clc; clear; close;
deff('z=f(x,y)','z=(y-x)/(y+x)')
x0=0;
y0=1; xn=0.1; h=0.02;
x=x0; y=y0;
while x~=xn
   y=y+h*f(x,y);
    x=x+h;
printf('\nWhen x=%g,y=%g\n',x,y);
end

disp('Khushi');
disp('03114813120');
```

OUTPUT:



EXPERIMENT - 10

AIM: Scilab code to solve ordinary differential equations using Runge-Kutta method.

Scilab Code:

```
//Range Kutta Method
clc
function ydot=f(x, y)
   ydot =x+y^2
   endfunction
x1=0;
y1=1;
h=0.1;
x(1)=x1;
y(1) = y1;
for i=1:2
k_1 = h*f(x(i),y(i));
k_2 = h*f(x(i)+0.5*h,y(i)+0.5*k_1);
k_3 = h*f((x(i)+0.5*h),(y(i)+0.5*k_2));
k_4 = h*f((x(i)+h),(y(i)+k_3));
k = (1/6)*(k_1 +2*k_2 +2*k_3 +k_4); y(i+1)= y(i)+ k;
printf('\n The value of y at x=%f is %f ', i*h,y(i+1))
x(i+1)=x(1)+ i*h;
end
disp('Khushi');
disp('03114813120');
```

OUTPUT:

```
The value of y at x=0.100000 is 1.116492
The value of y at x=0.200000 is 1.273563
Khushi

03114813120

-->
```