Roll. No. A016	Name: Varun Khadayate
Class B.Tech CsBs	Batch: 1
Date of Experiment: 29-07-2021	Subject: IT/WS

1. Create a 3 x 5 matrix of random real numbers. Delete the third row.

```
>> mat = rand(3,5)

mat =

   0.3238   0.7131   0.2204   0.7464   0.6632
   0.2118   0.2151   0.2079   0.4527   0.3126
   0.4927   0.4089   0.6518   0.8642   0.8317

>> mat(3,:) = []

mat =

   0.3238   0.7131   0.2204   0.7464   0.6632
   0.2118   0.2151   0.2079   0.4527   0.3126
```

2. Given the matrix:

```
>> mat = randi([1 20], 3,5) mat =
6     17    7     13    17
17    5      4      10     12
6     19     6      8      11

Why wouldn't this work:

mat(2:3, 1:3) = ones(2)
```

```
Because the left and right sides are not the same dimensions.
```

3. Create a three-dimensional matrix with dimensions 2 x 4 x 3 in which the first "layer" is all 0s, the second is all 1s and the third is all 5s. Use size to verify the dimensions.

```
>> mat3d = zeros(2,4,3)

mat3d(:,:,1) =

    0    0    0    0
    0    0    0    0


mat3d(:,:,2) =

    0    0    0    0
    0    0    0    0
```

```
mat3d(:,:,3) =

    0    0    0    0
    0    0    0    0

>> mat3d(:,:,2)   = 1

mat3d(:,:,1) =

    0    0    0    0
    0    0    0    0


mat3d(:,:,2) =

    1    1    1    1
    1    1    1    1


mat3d(:,:,3) =

    0    0    0    0
    0    0    0    0

>> mat3d(:,:,3)   = 5

mat3d(:,:,1) =

    0    0    0    0
    0    0    0    0


mat3d(:,:,2) =

    1    1    1    1
    1    1    1    1


mat3d(:,:,3) =

    5    5    5    5
    5    5    5    5

>> mat3d

mat3d(:,:,1) =
```

```
     0    0    0    0
     0    0    0    0


  mat3d(:,:,2) =

     1    1    1    1
     1    1    1    1


  mat3d(:,:,3) =

     5    5    5    5
   5    5    5    5
```

4. Create a vector x which consists of 20 equally spaced points in the range from $-\pi$ to $+\pi$. Create a y vector which is sin(x).

```
>> x = linspace(-pi,pi,20)

x =

  Columns 1 through 12

   -3.1416   -2.8109   -2.4802   -2.1495   -1.8188   -1.4881   -
1.1574   -0.8267   -0.4960   -0.1653    0.1653    0.4960

  Columns 13 through 20

    0.8267    1.1574    1.4881    1.8188    2.1495    2.4802
2.8109    3.1416

>> y = sin(x)

y =

  Columns 1 through 12

   -0.0000   -0.3247   -0.6142   -0.8372   -0.9694   -0.9966   -
0.9158   -0.7357   -0.4759   -0.1646    0.1646    0.4759

  Columns 13 through 20

    0.7357    0.9158    0.9966    0.9694    0.8372    0.6142
0.3247    0.0000
```

5. Create a 3 x 5 matrix of random integers, each in the inclusive range from -5 to 5. Get the sign of every element.

```
>> mat = randi([-5,5], 3,5)

mat =
```

```
    -2    -2     4     5    -2
     5    -3     2    -2     3
    -3     4     4     5    -4

>> sign(mat)

ans =

    -1    -1     1     1    -1
     1    -1     1    -1     1
    -1     1     1     1    -1
```

6. Find the sum 3+5+7+9+11.

```
>> sum(3:2:11)

ans =

    35
```

7. Find the sum of the first n terms of the harmonic series where n is an integer variable greater than one.

$$1+ 1/2+ 1/3+ 1/4+\ldots$$

```
>> n = 4

n =

    4

>> sum(1./(1:n))

ans =

2.0833
```

8. Find the following sum by first creating vectors for the numerators and denominators:

$$\frac{3}{1}+\frac{5}{2}+\frac{7}{3}+\frac{9}{4}$$

```
>> num = 3:2:9

num =

    3    5    7    9

>> denom = 1:4
```

```
denom =

     1     2     3     4

>> fracs = num ./ denom

fracs =

    3.0000    2.5000    2.3333    2.2500

>> sum(fracs)

ans =

   10.0833
```

9. Create a matrix and find the product of each row and column using prod.

```
>> mat = randi([1, 30], 2,3)

mat =

    13     9     9
    28    23     2

>> prod(mat)

ans =

   364   207    18

>> prod(mat,2)

ans =

      1053
      1288
```

10. Write a relational expression for a vector variable that will verify that the last value in a vector created by cumsum is the same as the result returned by sum.

```
>> vec = 2:3:17

vec =

    2     5     8    11    14    17

>> cv = cumsum(vec)

cv =

    2     7    15    26    40    57

>> sum(vec) == cv(end)

ans =

  logical

   1
```

11. Create a vector of five random integers, each in the inclusive range from -10 to 10. Perform each of the following:

```
>> vec = randi([-10, 10], 1,5)


vec =

   -3    10    -1     3     2
```

a. subtract 3 from each element

```
>> vec-3

ans =

    -6     7    -4     0    -1
```

b. count how many are positive

```
>> sum(vec > 0)

ans =

     3
```

c. get the cumulative minimum

```
>> cummin(vec)

ans =

   -3   -3   -3   -3   -3
```

12. Create a 3 x 5 matrix. Perform each of the following:

```
>> mat = randi([-10 10], 3,5)

mat =

     1     7    10    10    -7
    -4    -4     5    -2    10
    -5     6    -3     5     9
```

a. Find the maximum value in each column.

```
>> max(mat)

ans =

     1     7    10    10    10
```

b. Find the maximum value in each row.

```
>> max(mat, [], 2)

ans =

    10
    10
     9
```

c. Find the maximum value in the entire matrix.

```
>> max(max(mat))

ans =

    10
```

d. Find the cumulative maxima.

```
>> cummax(mat)

ans =

     1     7    10    10    -7
     1     7    10    10    10
     1     7    10    10    10
```

13. Find two ways to create a 3 x 5 matrix of all 100s (Hint: use ones and zeros).

```
>> ones(3,5)*100

ans =

   100   100   100   100   100
   100   100   100   100   100
   100   100   100   100   100

>> zeros(3,5)+100

ans =

   100   100   100   100   100
   100   100   100   100   100
   100 100   100   100   100
```

14. Given the two matrices:

```
    A         B
1   2   3     2   4   1
4   −1  6     1   3   0
```

Perform the following operations:

```
>> A = [1,2,3;4,-1,6]

A =

     1     2     3
     4    -1     6

>> B = [2,4,1;1,3,0]

B =

     2     4     1
     1     3     0
```

a. A + B

```
>> A + B

ans =

     3     6     4
     5     2     6
```

b. A - B

```
>> A - B

ans =

    -1    -2     2
     3    -4     6
```

c. A * B

```
>> A .* B

ans =

     2     8     3
     4    -3     0
```

15. The built-in function clock returns a vector that contains 6 elements: the first three are the current date (year, month, day) and the last three represent the current time in hours, minutes, and seconds. The seconds is a real number, but all others are integers. Store the result from clock in a variable called myc. Then, store the first three elements from this variable in a variable today and the last three elements in a variable now. Use the fix function on the vector variable now to get just the integer part of the current time.

```
>> myc = clock

myc =

   1.0e+03 *

    2.0220    0.0070    0.0290    0.0160    0.0030    0.0567

>> today = myc(1:3)

today =

        2022           7          29

>> now = myc(4:end)

now =

   16.0000    3.0000   56.7430

>> fix(now)

ans =

    16     3    56
```

16. A vector v stores for several employees of the Green Fuel Cells Corporation their hours worked one week followed for each by the hourly pay rate. For example, if the variable stores

```
>> v
v =
      33.0000 10.5000 40.0000 18.0000 20.0000 7.5000
```

that means the first employee worked 33 hours at $10.50 per hour, the second worked 40 hours at $18 an hour, and so on. Write code that will separate this into two vectors, one that stores the hours worked and another that stores the hourly rates. Then, use the array multiplication operator to create a vector, storing in the new vector the total pay for every employee.

```
>> v = [33.0000 10.5000 40.0000 18.0000 20.0000 7.5000]

v =

  33.0000   10.5000   40.0000   18.0000   20.0000    7.5000

>> hours = v(1:2:length(v))

hours =

   33    40    20

>> payrate = v(2:2:length(v))

payrate =

  10.5000   18.0000    7.5000

>> totpay = hours .* payrate

totpay =

 346.5000  720.0000  150.0000
```

17. A company is calibrating some measuring instrumentation and has measured the radius and height of one cylinder 10 separate times; they are in vector variables r and h. Find the volume from each trial, which is given by Πr2h. Also use logical indexing first to make sure that all measurements were valid (> 0).

```
>> r = [5.501 5.5 5.499 5.498 5.5 5.5 5.52 5.51 5.5 5.48];
>> h = [11.11 11.1 11.1 11.12 11.09 11.11 11.11 11.1 11.08 11.11];
```

```
>> r = [5.501 5.5 5.499 5.498 5.5 5.5 5.52 5.51 5.5 5.48]

r =

    5.5010    5.5000    5.4990    5.4980    5.5000    5.5000
5.5200    5.5100    5.5000    5.4800

>> h = [11.11 11.1 11.1 11.12 11.09 11.11 11.11 11.1 11.08 11.11]

h =

   11.1100   11.1000   11.1000   11.1200   11.0900   11.1100
11.1100   11.1000   11.0800   11.1100

>> vol = pi * r.^2 .* h

vol =

   1.0e+03 *

    1.0562    1.0549    1.0545    1.0560    1.0539    1.0558
1.0635    1.0587    1.0530    1.0482
```

18. For the following matrices A, B, and C:

$$A=\begin{bmatrix}1 & 4\\ 3 & 2\end{bmatrix}\quad B=\begin{bmatrix}2 & 1 & 3\\ 1 & 5 & 6\\ 3 & 6 & 0\end{bmatrix}\quad C=\begin{bmatrix}3 & 2 & 5\\ 4 & 1 & 2\end{bmatrix}$$

```
>> A = [1 4;3,2]

A =

     1     4
     3     2

>> B = [2 1 3;1 5 6;3 6 0]

B =

     2     1     3
     1     5     6
     3     6     0

>> C = [3 2 5;4 1 2]

C =

     3     2     5
     4     1     2
```

a. Give the result of 3*A.

```
>> 3 * A

ans =

     3    12
     9     6
```

b. Give the result of A*C.

```
>> A * C

ans =

    19     6    13
    17     8    19
```

c. Are there any other matrix multiplications that can be performed? If so, list them.

```
>> C * B

ans =

    23    43    21
    15    21    18
```

19. For the following vectors and matrices, A, B, and C:

$$A=\begin{bmatrix}4 & 1 & -1\\ 2 & 3 & 0\end{bmatrix}\quad B=[1\ \ 4]\quad C=\begin{bmatrix}2\\ 3\end{bmatrix}$$

```
>> A = [4 1 -1;2 3 0]

A =

     4     1    -1
     2     3     0

>> B = [1 4]

B =

     1     4

>> C = [2;3]

C =

     2
     3
```

a. A * B

```
>> A * B
Error using  *
Incorrect dimensions for matrix multiplication. Check that the
number of columns in the first matrix matches the number
of rows in the second matrix. To operate on each element of the
matrix individually, use TIMES (.*) for elementwise
multiplication.
```

b. B * C

```
>> B * C

ans =

    14
```

c. C * B

```
>> C * B

ans =

     2     8
     3    12
```

20. The matrix variable rainmat stores the total rainfall in inches for some districts for the years 2010-2013. Each row has the rainfall amounts for a given district. For example, if rainmat has the value:

```
>> rainmat
ans =
   25    33    29    42
```

```
   53    44    40    56
   etc.
```

district 1 had 25 inches in 2010, 33 in 2011, etc. Write expression(s) that will find the number of the district that had the highest total rainfall for the entire four year period.

```
>> rainmat = [25 33 29 42; 53 44 40 56]

rainmat =

   25   33   29   42
   53   44   40   56

>> large = max(max(rainmat))

large =

   56

>> linind = find(rainmat== large)

linind =

    8

>> floor(linind/4)

ans =

    2
```

21. Generate a vector of 20 random integers, each in the range from 50 to 100. Create a variable evens that stores all of the even numbers from the vector, and a variable odds that stores the odd numbers.

```
>> nums = randi([50, 100], 1, 20)

nums =

    99    95    92    90    70    87    74    72    93    80    59
55    61    58    71    52    77    65    86    56

>> evens = nums(rem(nums,2)==0)

evens =

    92    90    70    74    72    80    58    52    86    56

>> odds = nums(rem(nums,2)~=0)

odds =

    99    95    87    93    59    55    61    71    77    65
```

22. Assume that the function diff does not exist. Write your own expression(s) to accomplish the same thing for a vector.

```
>> vec = [5 11 2 33 -4]

vec =

     5    11     2    33    -4

>> v1 = vec(2:end)

v1 =

    11     2    33    -4

>> v2 = vec(1:end-1)

v2 =

     5    11     2    33

>> v1-v2

ans =

     6    -9    31   -37
```

23. Create a vector variable vec; it can have any length. Then, write assignment statements that would store the first half of the vector in one variable and the second half in another. Make sure that your assignment statements are general, and work whether vec has an even or odd number of elements (Hint: use a rounding function such as fix).

```
>> vec = 1:9

vec =

   1   2   3   4   5   6   7   8   9

>> fhalf = vec(1:fix(length(vec)/2))

fhalf =

   1   2   3   4

>> shalf = vec(fix(length(vec)/2)+1:end)

shalf =

   5   6   7   8   9
```