LAB 2 - Trevor Callow - MAT 275

Exercise 1

Part (a)

```
A=[2,7,-1;4,5,-3;-2,-3,5];
B=[12,19,-7;3,14,2;11,11,8];
b=[8;-17;-7];
c=[4,-3,4];
d=[-3;3;1];
AB = A*B;
BA = B*A;
cA = c*A;
Bd = B*d;
```

Part (b)

```
C=[A;B]
```

```
C = 6×3
     2     7    -1
     4     5    -3
    -2    -3     5
    12    19    -7
     3    14     2
    11    11     8
```

```
D=[B,d]
```

```
D = 3×4
    12    19    -7    -3
     3    14     2     3
    11    11     8     1
```

Part (c)

NOTE: *Use the 'backslash' command, NOT the division operator '/' .Delete these notes before submission.*

```
x = A\b
```

```
x = 3×1
  -12.2500
    4.0938
   -3.8438
```

Part (d)

```
A(3,3)=0;
```

Part (e)

```
a=A(2,:);
```

Part (f)

```
B(:,3)=[]
```

```
B = 3×2
    12    19
     3    14
    11    11
```

Exercise 2

Part (a)

NOTE: *We must create separate functioin M-files here. The function will require input variables. Therefore, you cannot use 'run' to execute the function. You must invoke it by providing values for the input variables. Name the function geomsum1, then complete the follwing.*

Display contents of geomsum1 M-file

```
type 'geomsum1.mlx'
```

```
function Z =  geomsum1(r,a,n)
Z = 0;
for i=0:n-1
    Z=Z+a*r^i;
end
end
```

Assign values to input variables

```
r = (-9)/11;
a = 1;
n = 10;
```

NOTE: *Do NOT define r,a, or n inside the function file. This defeats the purpose of requiring input arguments. Delete this note before submitting.*

Compute geometric sum for specified values of r,a, and n.

```
Z = geomsum1(r,a,n) ;
```

Part (b)

NOTE: *MATLAB has several built in functions. Here, we want to use the built-in function "sum" to compute the same geometric sum as computed in part (a). The "sum" function takes a vector as input and sums its elements. Hence, we must create a vector containing all the terms of the sum we wish to compute and pass that vector as input to the sum function. Type "help sum" in command window for more info. Don't forget: the name of your function should be geomsum2. Delete this note before submission.*

```
type 'geomsum2.mlx'
```

```
function Z = geomsum2(r,a,n)
e=0:n-1;
R=r.^e;
Z=sum(a*R);
end
```

```
r=(-9)/11
```

```
r = -0.8182
```

```
a=1
```

```
a = 1
```

```
n=10
```

```
n = 10
```

Exercise 3

Part (a)

NOTE: *When the protocol says to write a script file, you do not need to create a separate M-file. This main file is a script file and you can simply write the list of commands here. Delete this note before submission.*

Initiate product P.

```
P = 1;
```

Define starting iteration index.

```
m = 2;
```

Define stepsize of iteration.

```
k = 2;
```

Define ending iteration index.

```
n = 14;
```

Compute product.

```
for i = m:k:n
    P = P*i  % muliply P by next element at each iteration (suppress output)
end
```

```
P = 2
P = 8
P = 48
P = 384
P = 3840
P = 46080
P = 645120
```

Display product.

```
disp(P)
```

```
      645120
```

NOTE: *i = m:k:n is also a vector! A for-loop essentially iterates through each element of a vector. Keep this in mind for part (b).*

Part (b)

NOTE*: Emphasis on the part where it says "SINGLE COMMAND." That means ONE line of code. The "prod" function is another built-in function that, similar to "sum", takes in a vector as input, yet computes the product of all the elements of the input vector. So, we need to create a vector containing all the numbers between 1 and 15 with a stepsize of 2. Which vector declaration method is best when we know the stepsize? Also, in order to compute the product in a single command, we must declare the vector INSIDE the prod command as an input argument, as opposed to x =vector, prod(x). Instead, we do prod(vector). Type "help prod" in the command window for more info. Delete this note before submission.*

```
P = prod(2:2:14);
disp(P);
```

```
      645120
```

Exercise 4

NOTE: *When the protocol says to write a script file, you do not need to create a separate M-file. This main file is a script file and you can simply write the list of commands here. Refer to second example on page 7 of protocol before attempting this exercise. Delete this note before submission.*

Initiate variables.

```
power = 3;
k = 1; % initiate counter
```

Initialize the vector v to the empty vector

```
v = [];
```

Compute powers and store in v.

```
while power < 10^5   % specify condition of while-loop: stop iterating once
                % condition is no longer satisfied
    v(k) = power;     % evaluate kth entry of the vector  v
    k = k+1;     % increment counter k
    power = power * 3; % compute next value of power at each iteration
end
```

Display vector v.

```
disp(v);
```

```
           3           9          27          81         243         729        2187        6561       196
```

Exercise 5

***NOTE**: Here, you must create a separate function M-file f that accepts one input argument, a value of the variable x. You may want to start creating a new folder for each lab as we will be using f multiple times to define different functions. Please refer to the example on the last page of the protocol before attempting this problem. Delete this note before submission.*

Display contents of function f M-file.

```
type 'f.mlx'
```

```
function result = f(x)

if x <= 3
    result=exp(x-7);
elseif x>3 && x <= 6
        result = x-3;
elseif x>6 && x~=7
        result = x / (x-7);
elseif x == 7
        disp('the function is undefined at x = 7');
        result = NaN;
end
end
```

Evaluate f at the given vaue of x.

```
disp(f(2))
```

```
    0.0067
```

Evaluate f at the given value of x.

```
disp(f(3))
```

```
    0.0183
```

% Evaluate f at the given value of x.

```
disp(f(3.5))
```

```
    0.5000
```

Evaluate f at the given value of x.

```
disp(f(6))
```

```
     3
```

Evaluate f at the given value of x.

```
disp(f(7))
```

```
the function is undefined at x = 7
   NaN
```

Evaluate f at the given value of x.

```
disp(f(8))
```

8