VECTOR OPERATIONS

Q1) Generate two vectors u and v by typing the code given below.

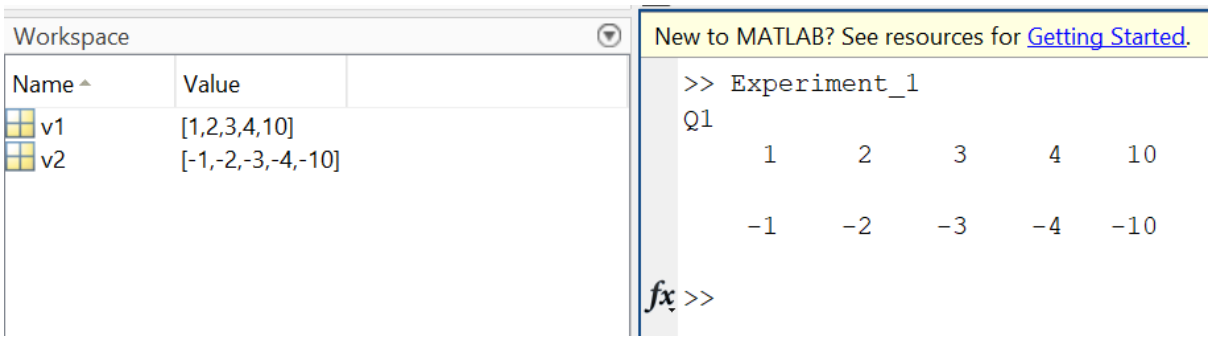
v1 = [1 2 3 4 10]

v2 = [−1 − 2 − 3 − 4 − 10]

Code:

```
%Q1
disp("Q1");
v1=[1 2 3 4 10];
v2=[-1 -2 -3 -4 -10];
disp(v1);
disp(v2);
```

Output:

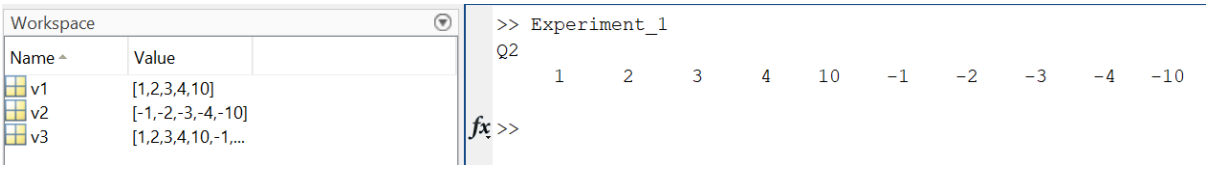


Q2) Concatenate two vectors to form a new vector v3. Use v3 = [u, v] or v3 = [u v]

Code:

```
%Q2
disp("Q2");
v3=[v1 v2];
disp(v3);
```

Output:

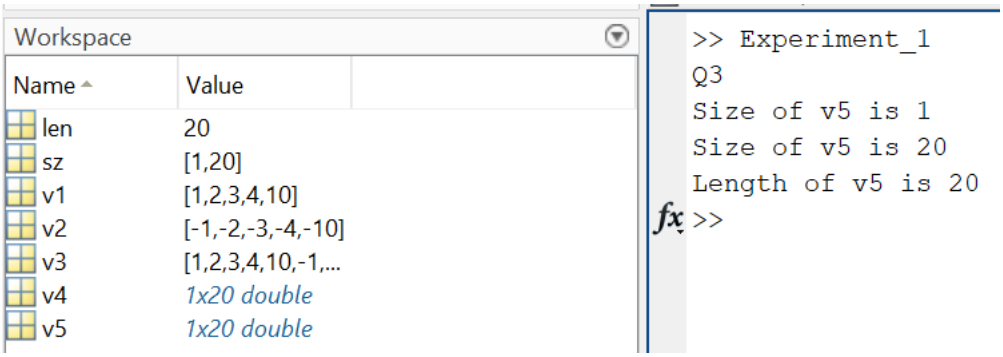


Q3) Type help ones and help zeros. Create vectors v4 of 20 zeros and v5 of 20 ones using the command. Use the size or length command to identify the size of the created vector.

Code:

```
%Q3
disp("Q3");
%help ones;
%help zeros;
v4=zeros(1,20);
v5=ones(1,20);
%help size;
sz=size(v5);
len=length(v5);
fprintf("Size of v5 is %d\n",sz);
fprintf("Length of v5 is %d\n",len);
```

Output:



Q4) Create a vector v6 of length 20 with every entry equal to 5.

Code:

```
%Q4
disp("Q4");
v6=5*ones(1,20);
disp(v6);
```

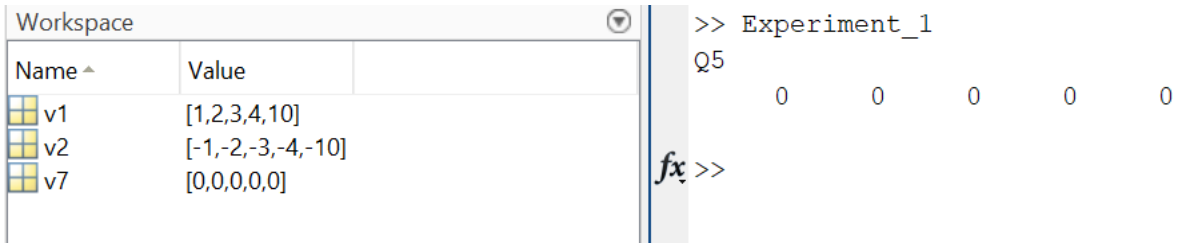
Output:

Q5) Add two vectors v1 and v2 by using v1 + v2. Assign it to v7.

Code:

```
%Q5
disp("Q5");
v7=v1+v2;
disp(v7);
```

Output:

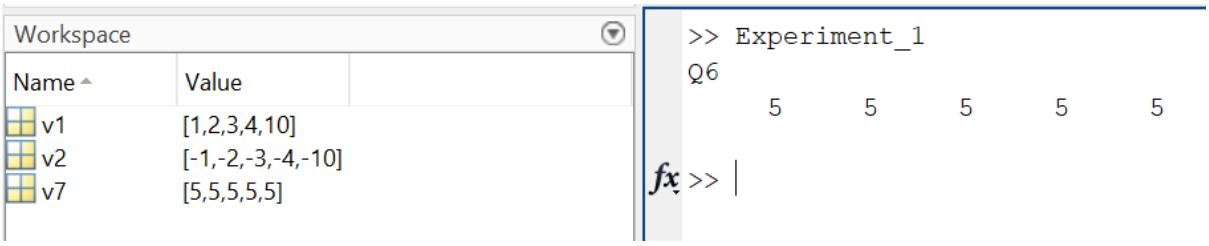


Q6) Add a constant to all entries of v7 vector. Try v7 + 5

Code:

```
%6
disp("Q6");
v7=v7+5;
disp(v7);
```

Output:



Q7) Try 1:2:10. Observe the result. Use this to create vectors

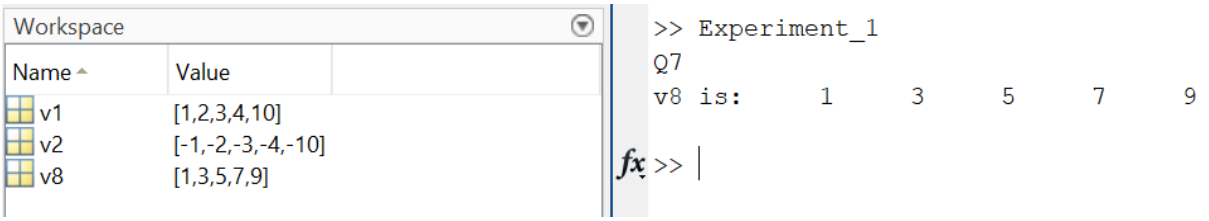
v8 = [1 2 3 . . . 10]

v9 = [25 20 15 . . . – 20]

Code:

```
%7
disp("Q7");
v8=1:2:10;
fprintf("v8 is:");
disp(v8);
```

Output:

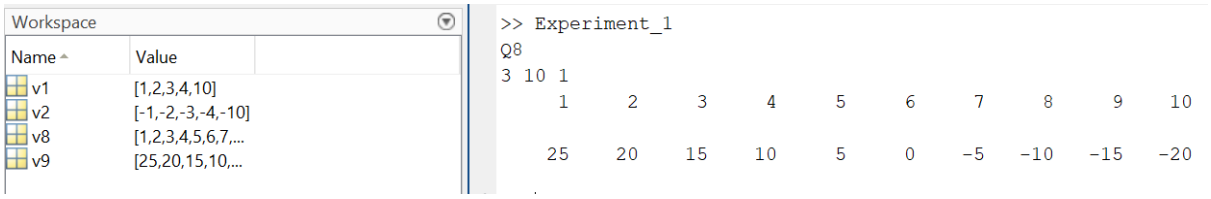


Q8) Pick the third value in vector v1. Use v1(3). Note that the numbering of entries is from 1 to length(v1). Try v1(6) and v1(0).

Code:

```
%8
disp("Q8");
fprintf("%d %d %d\n",v1(3),v1(5),v1(1));
v8=1:1:10;
v9=25:-5:-20;
disp(v8);
disp(v9);
```

Output:

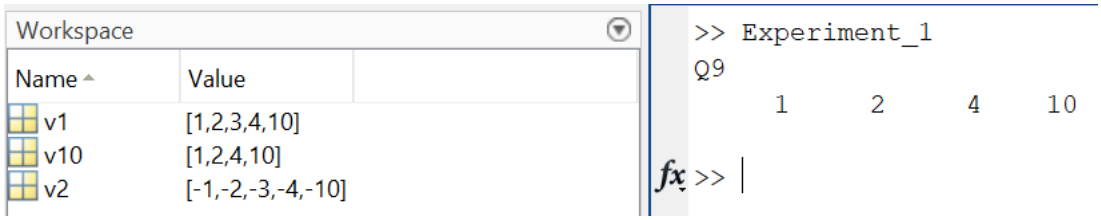


Q9) Obtain v10 from v1 by removing the third entry.

Code:

```
%9
disp("Q9");
v10=v1;
v10(3)=[];
disp(v10);
```

Output:

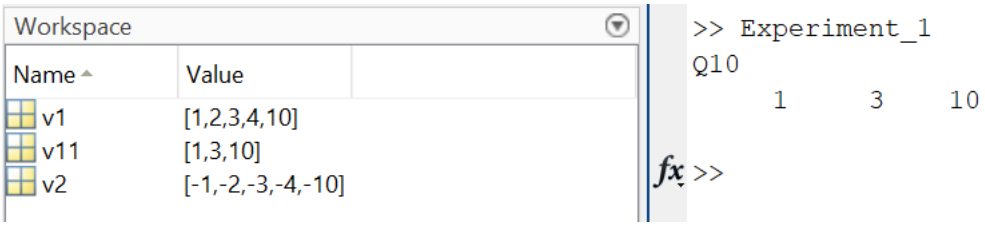


Q10) Obtain v11 from v1 by picking only the odd indexed entries.

Code:

```
%10
disp("Q10");
v11=v1(1:2:length(v1));
disp(v11);
```

Output:



Q11) Obtain the transpose of v1 and save it as v1t .

Code:

```
%11
disp("Q11");
v1t=v1';
disp(v1t);
```

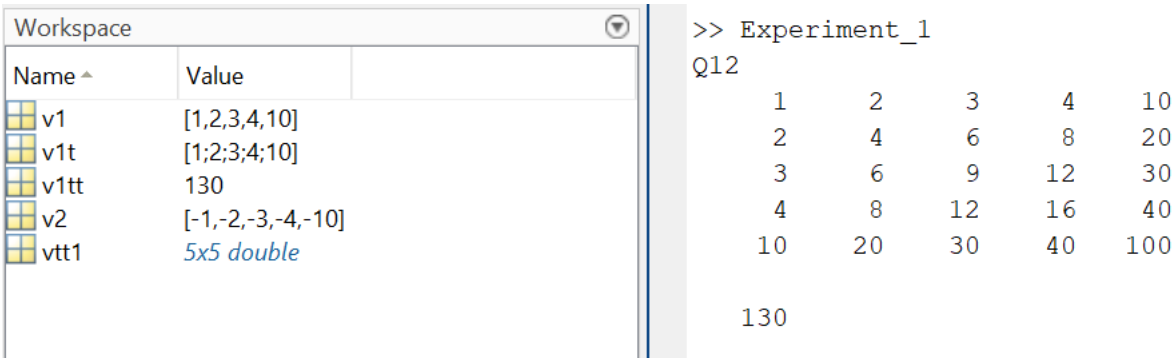
Output:

Q12) Try the operations v1 ∗ v1t and v1t ∗ v1

Code:

```
%12
disp("Q12");
vtt1=v1t*v1;
v1tt=v1*v1t;
disp(vtt1);
disp(v1tt);
```

Output:

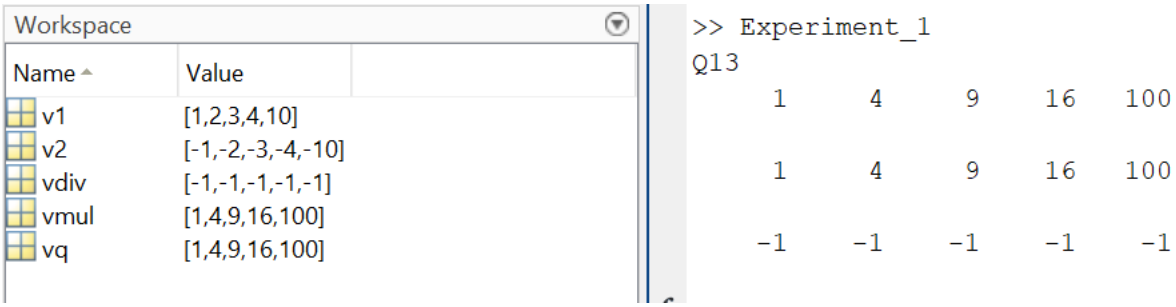


Q13) Try the operation v1. ∗ v1. Also try v1.∧v2 and v1./v2.

Code:

```
%13
disp("Q13");
vq=v1.*v1;
disp(vq);
vmul=v1.^2;
disp(vmul);
vdiv=v1./v2;
disp(vdiv);
```

Output:

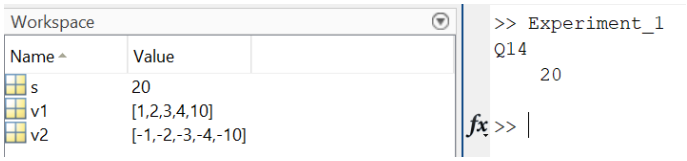


Q14) Add all the entries of v1 and store it in s

Code:

```
%14
disp("Q14");
s=sum(v1);
disp(s);
```

Output:

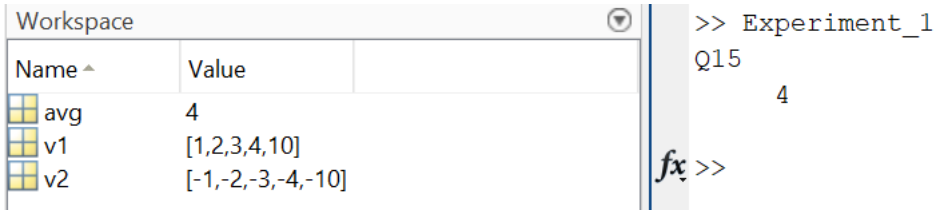


Q15) Find the average of entries in v1

Code:

```
%15
disp("Q15");
avg=mean(v1);
disp(avg);
```

Output:

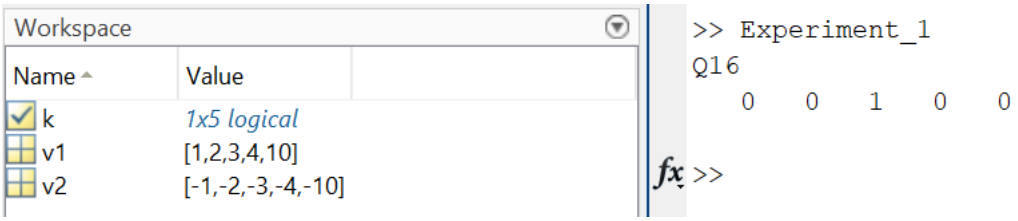


Q16) Try v1 == 3. Observe the result.

Code:

```
%16
disp("Q16");
k=v1==3;
disp(k);
```

Output:

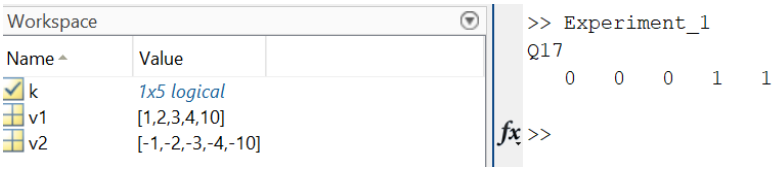


Q17) Find the number of entries in v9 greater than or equal to 15. Do not use any loops.

Code:

```
%17
disp("Q17");
k=v1>=4;
disp(k);
```

Output:

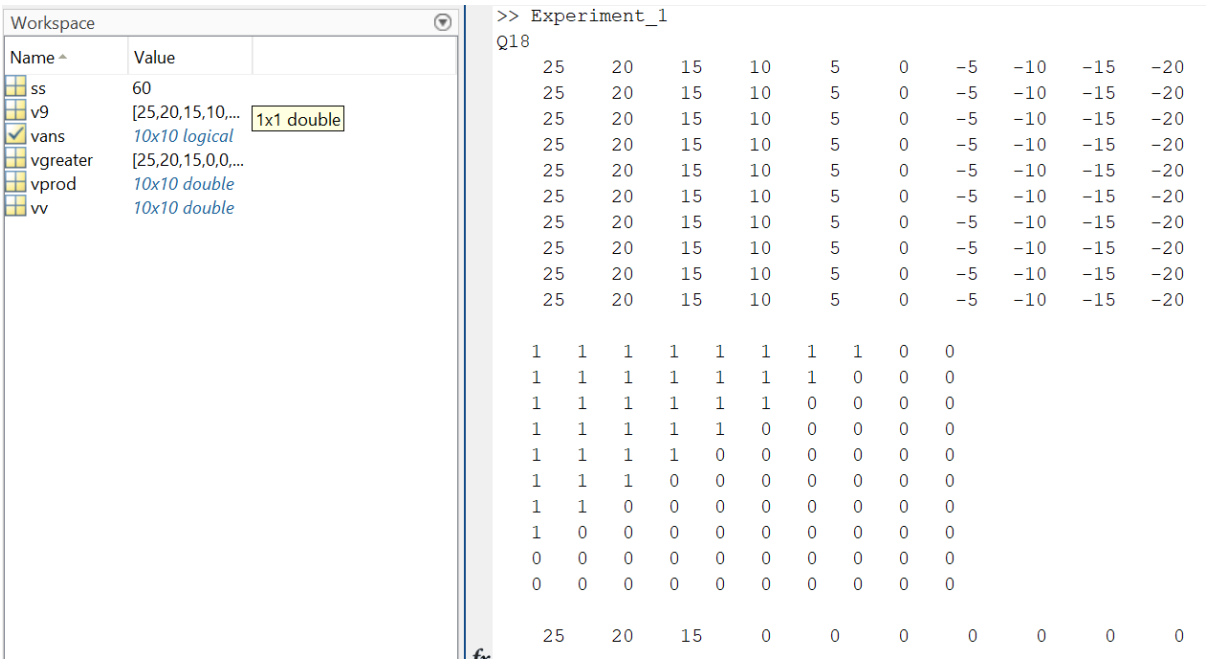


Q18) Provide two different methods without using loops to find the sum of entries in v9 which are greater than 15. Hint: Use '∗' and '.∗'

Code:

```
%18
disp("Q18");
vprod=ones(length(v9),1)*v9;
disp(vprod);
vv=vprod+vprod';
vans=vv>=15;
disp(vans);

vgreater=(v9>=15).*v9;
ss=sum(vgreater);
disp(vgreater);
disp(ss);
```

Output:



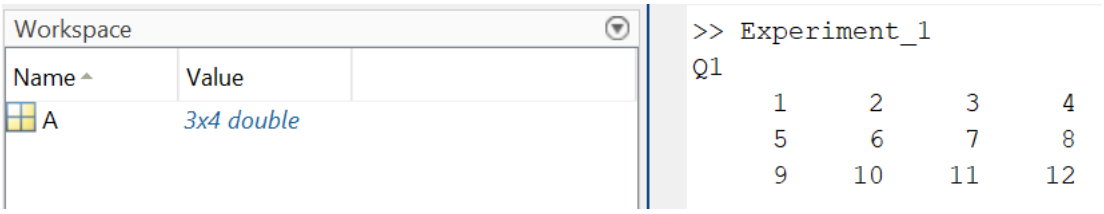
MATRIX OPERATIONS

Q1) Create a 3 × 4 matrix A. Use A = [1 2 3 4; 5 6 7 8; 9 10 11 12];

Code:

```
%1
disp("Q1");
A = [1 2 3 4; 5 6 7 8; 9 10 11 12];
disp(A);
```

Output:

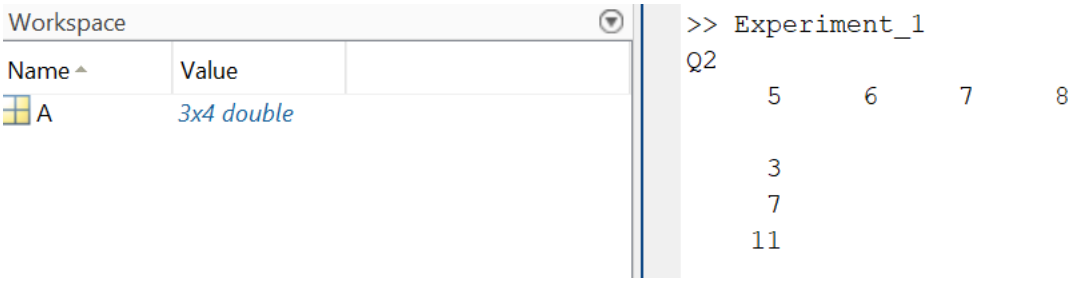


Q2) Print only the second row of A using A(2, :) and print only the third column of A using A(:, 3).

Code:

```
%2
disp("Q2");
disp(A(2,:));
disp(A(:,3));
```

Output:

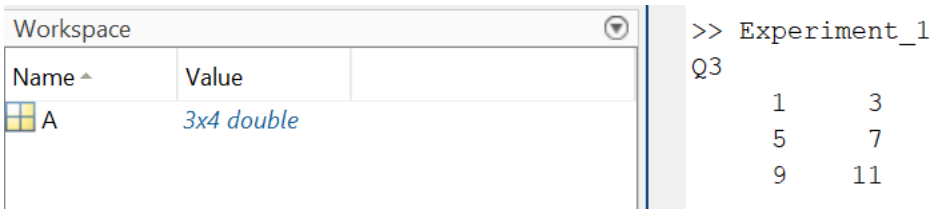


Q3) Print only first and third column of A.

Code:

```
%3
disp("Q3");
disp([A(:,1) A(:,3)]);
```

Output:

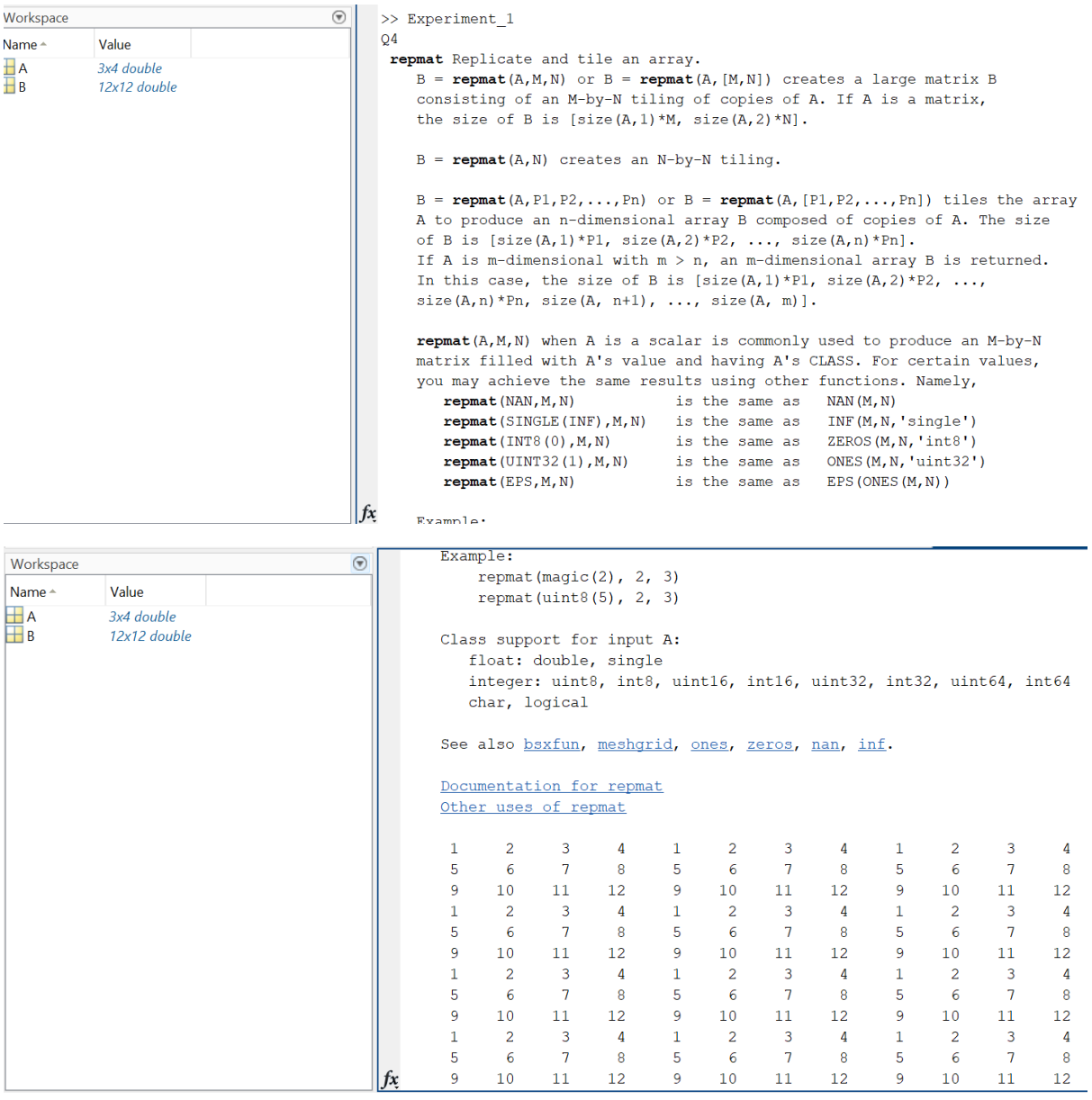


Q4) Type help repmat. Use repmat command to create a 12 × 12 matrix B repeating the matrix A.

Code:

```
%4
disp("Q4");
help repmat;
B=repmat(A,[4,3]);
disp(B);
```

Output:

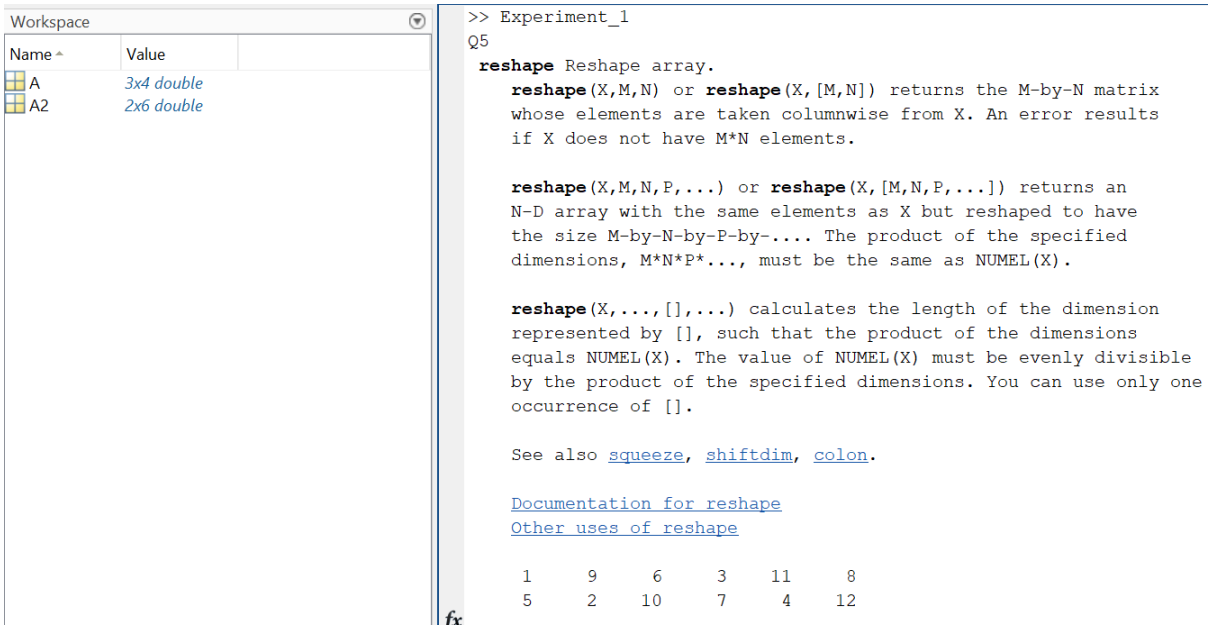


Q5) Use reshape command to change A into 2 × 6 matrix.

Code:

```
%5
disp("Q5");
help reshape;
A2=reshape(A,[2,6]);
disp(A2);
```

Output:

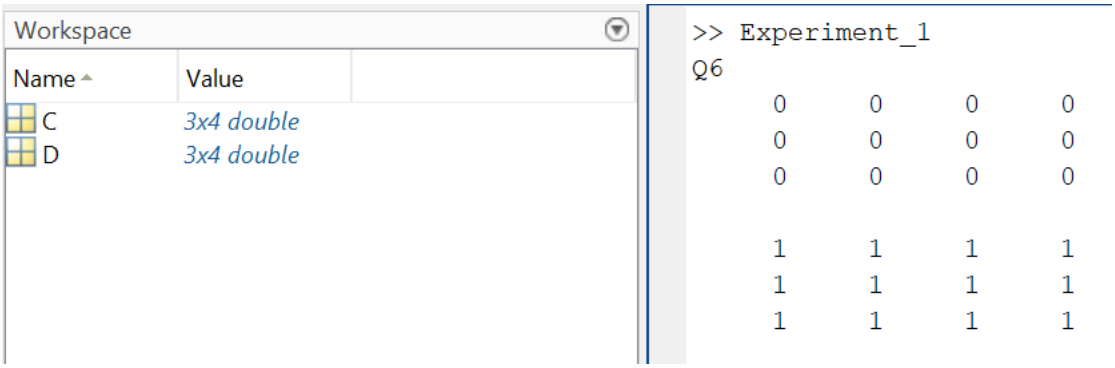


Q6) Use zeros, ones commands to create matrices C and D both of order 3 × 4.

Code:

```
%6
disp("Q6");
C=zeros(3,4);
D=ones(3,4);
disp(C);
disp(D);
```

Output:

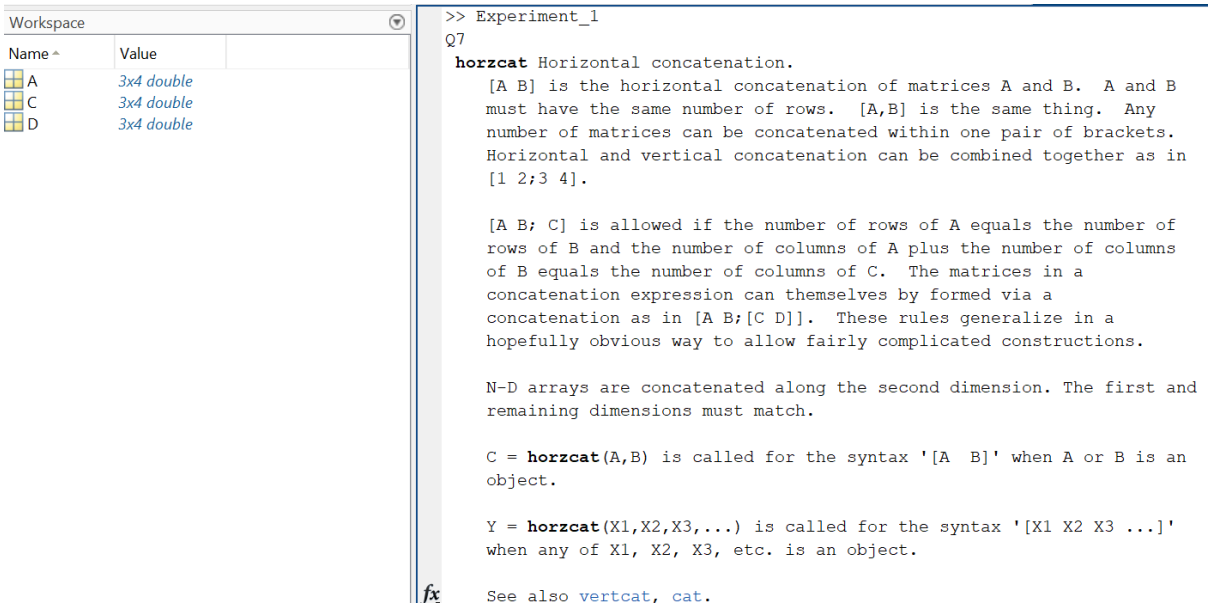


Q7) Use horzcat to concatenate A and C and use vertcat to concatenate A and D.

Code:

```
%7
disp("Q7");
help horzcat;
disp(horzcat(A,C));
disp(vertcat(A,D));
```

Output:



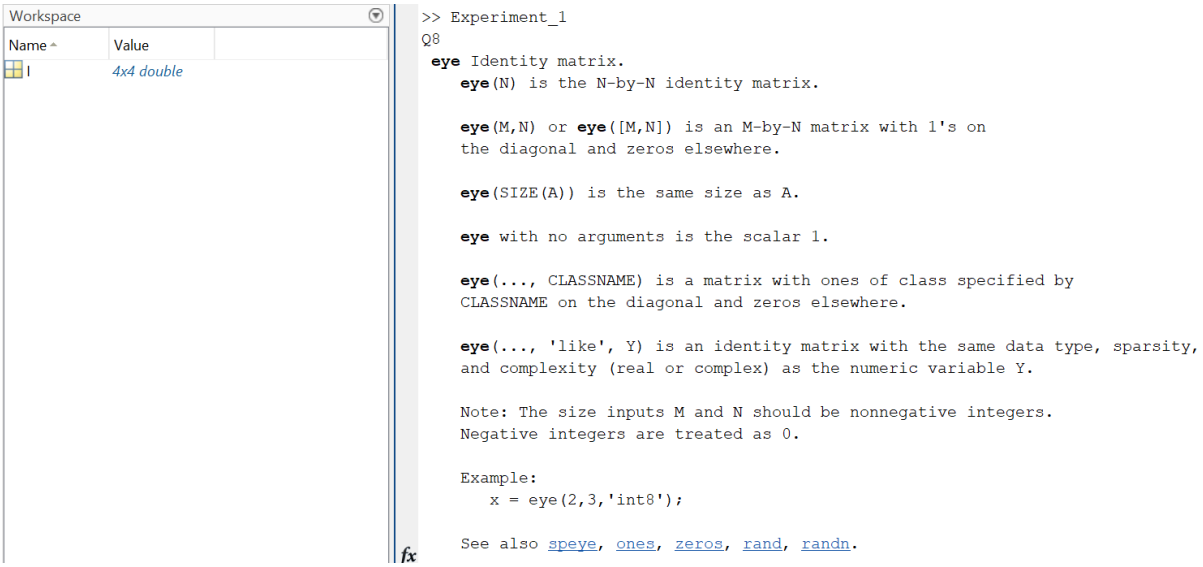

```
     1     2     3     4     0     0     0     0
     5     6     7     8     0     0     0     0
     9    10    11    12     0     0     0     0

     1     2     3     4
     5     6     7     8
     9    10    11    12
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

Q8) Create an identity matrix using eye command.

Code:

```
%8
disp("Q8");
help eye;
I=eye(4);
disp(I);
```

Output:



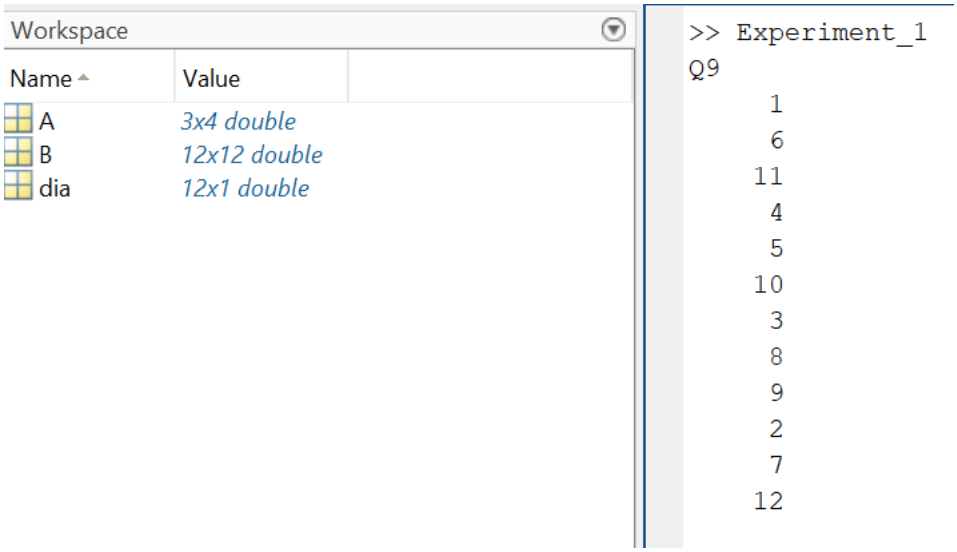

```
1     0     0     0
0     1     0     0
0     0     1     0
0     0     0     1
```

Q9) Use diag command to obtain diagonal elements of B.

Code:

```
%9
disp("Q9");
dia=diag(B);
disp(dia);
```

Output:

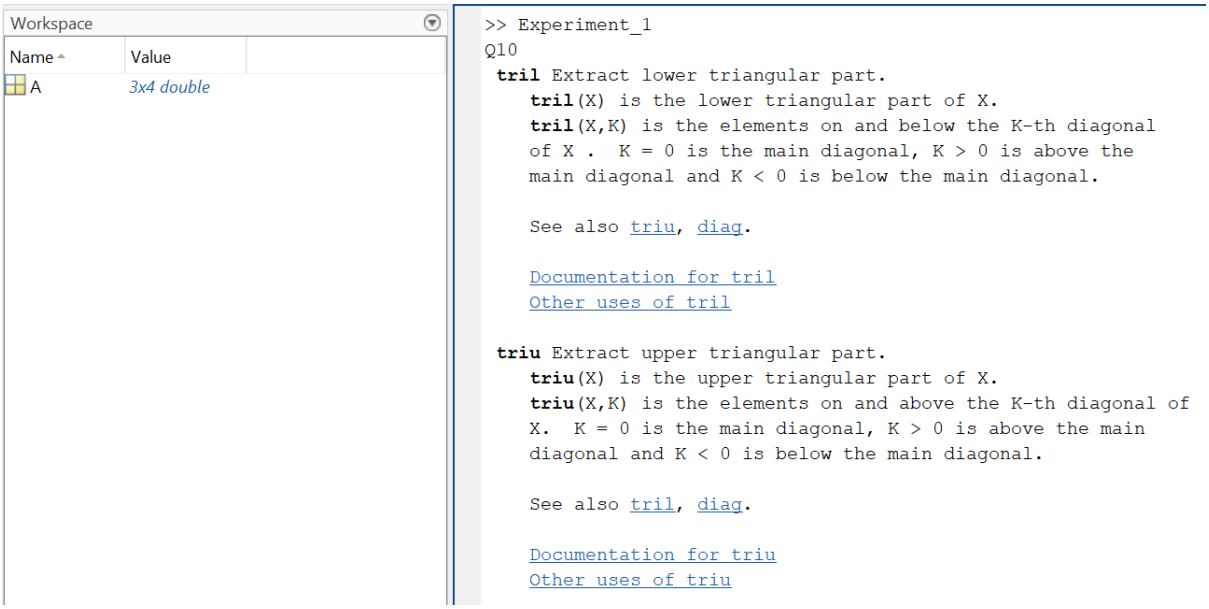


Q10) Use tril and triu commands on the matrix B. Apply these on matrix A also.

Code:

```
%10
disp("Q10");
help tril;
help triu;
disp(tril(A));
disp(triu(A));
```

Output:



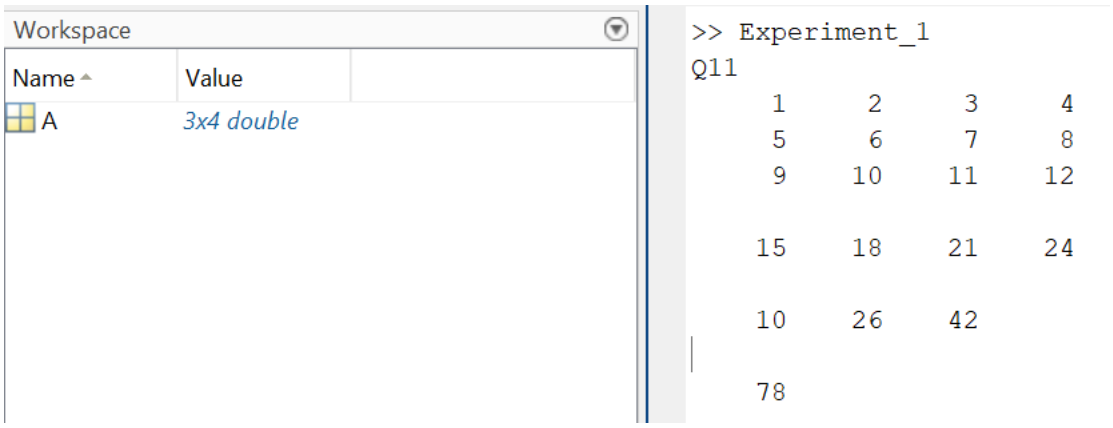

```
     1     0     0     0
     5     6     0     0
     9    10    11     0

     1     2     3     4
     0     6     7     8
     0     0    11    12
```

Q11) Using sum command find the sum of each row and the sum of each column of matrix A. Find the sum of all entries in the matrix. Use sum(A(:)) or ‘all’.

Code:

```
%11
disp("Q11");
disp(A);
disp(sum(A));
disp(sum(A'));
disp(sum(A(:)));
```

Output:

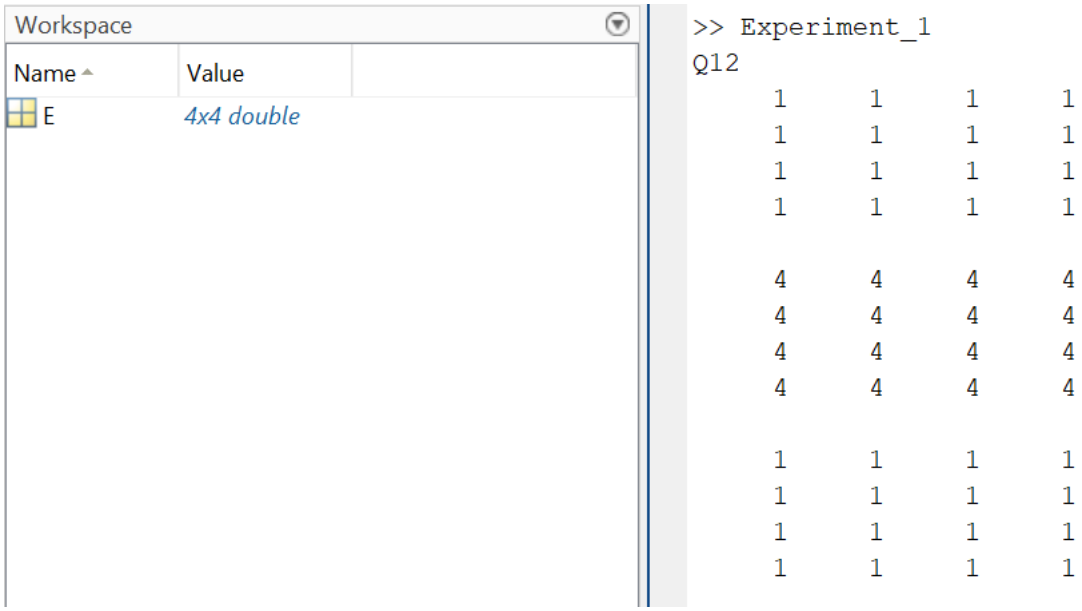


Q12) Understand the difference between '∗' and '.∗' in matrix multiplication by choosing suitable matrices.

Code:

```
%12
disp("Q12");
E=ones(4);
disp(E);
disp(E*E);
disp(E.*E);
```

Output:



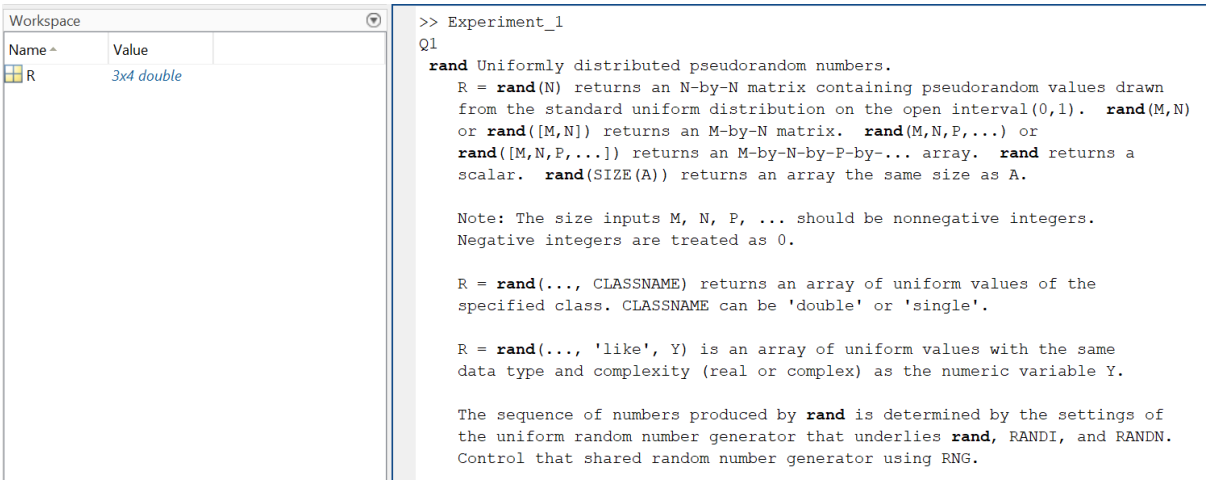
RANDOM MATRICES GENERATION

Q1) Use rand command to generate a random matrix R of order 3×4.

Code:

```
%1
disp("Q1");
help rand;
R=rand([3,4]);
disp(R);
```

Output:



See Replace Discouraged Syntaxes of rand and randn to use RNG to replace **rand** with the 'seed', 'state', or 'twister' inputs.

See also randi, randn, rng, RandStream, RandStream/rand, sprand, sprandn, randperm.

Documentation for rand
Other uses of rand

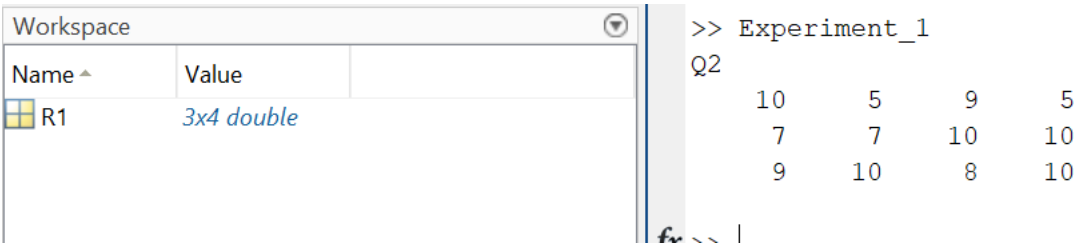
```
0.8147    0.9134    0.2785    0.9649
0.9058    0.6324    0.5469    0.1576
0.1270    0.0975    0.9575    0.9706
```

Q2) Note that all the random values generated lie between 0 and 1. Suitably modify the code to obtain random matrices with entries between 5 and 10.

Code:

```
%2
disp("Q2");
R1=randi([5,10],[3,4]);
disp(R1);
```

Output:

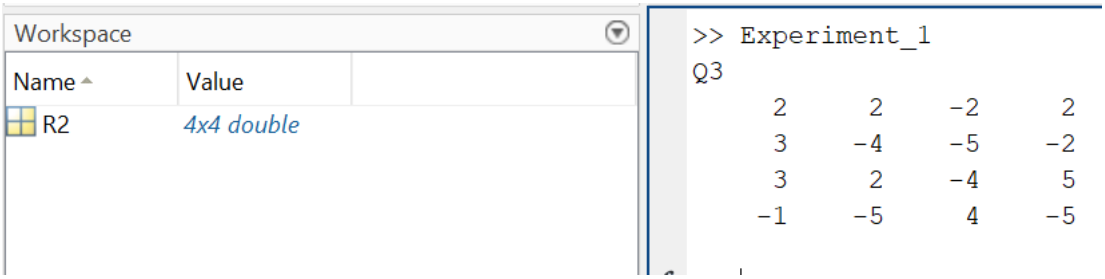


Q3) Use randi command to generate a matrix of uniformly distributed integers. The integers generated should belong to the interval −5 to 5.

Code:

```
%3
disp("Q3");
R2=randi([-5,5],4);
disp(R2);
```

Output:

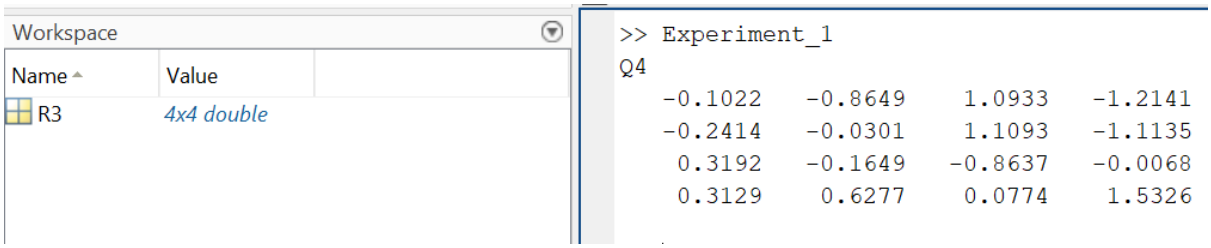


Q4) Use randn command to generate a matrix of random values following standard normal distribution.

Code:

```
%4
disp("Q4");
R3=randn(4);
disp(R3);
```

Output:

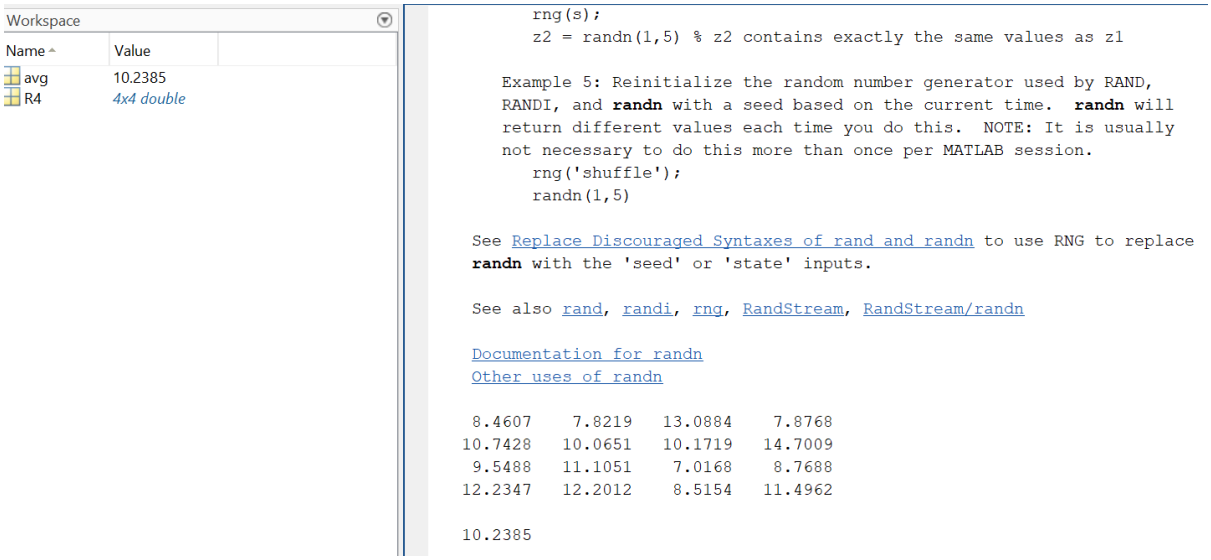


Q5) Modify the code to obtain entries from a normal distribution with mean 10 and variance 4.

Code:

```
%5
disp("Q5");
help randn;
R4=10+2.*randn(4);
disp(R4);
avg=mean(R4(:));
disp(avg);
```

Output:

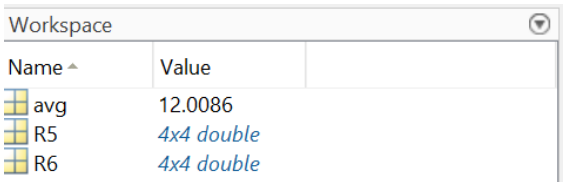


Q6) Do the same for Rayleigh and Exponential distribution with scale parameter = 2 and mean parameter = 10 respectively.

Code:

```
%6
disp("Q6");
R5=10+raylrnd(2,4);
disp(R5);
avg=mean(R5(:));
disp(avg);
R6=10+exprnd(2,4);
disp(R6);
avg=mean(R6(:));
disp(avg);
```

Output:




```
>> Experiment_1
Q6
   10.5781   13.1905   12.8434   12.3267
   12.9318   15.2624   13.9098   12.2573
   12.7588   11.3802   10.9630   11.6821
   12.8124   10.5423   14.7924   11.1932

   12.4640

   11.0151   10.5803   10.1813   10.0077
   12.6714   11.5946   13.7628   15.0976
   10.8491   14.9581   10.3828   11.6298
   10.7444   12.9483   11.2385   14.4764

   12.0086
```

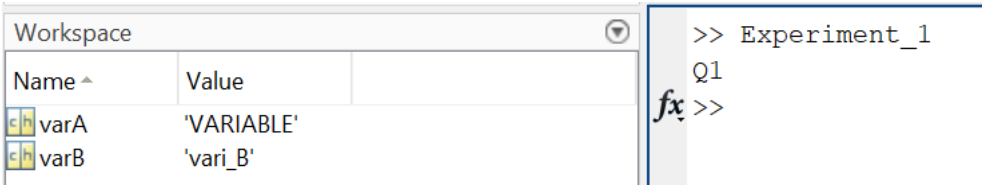
SAVING AND LOADING

Q1) Create and save two variables A and B to a file savefile.mat

Code:

```
%1
disp("Q1");
varA='VARIABLE';
varB='vari_B';
save("savefile.m.mat",'varA');
save("savefile.m.mat",'varB');
```

Output:

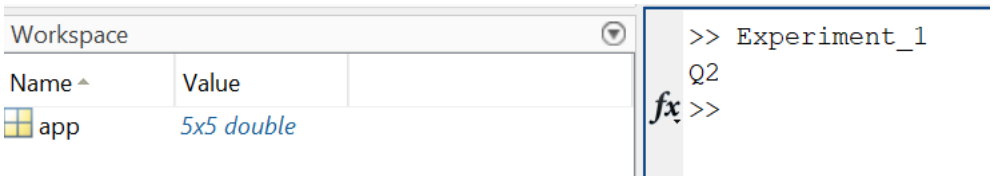


Q2) Create a third variable C and append it to savefile.mat

Code:

```
%2
disp("Q2");
app=ones(5,5);
save("savefile.m.mat","app","-append");
```

Output:

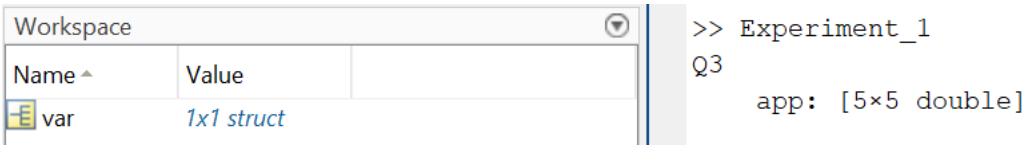


Q3) Load all three variables from savefile.mat

Code:

```
%3
disp("Q3");
var=load("savefile.m.mat","app");
disp(var);
```

Output:

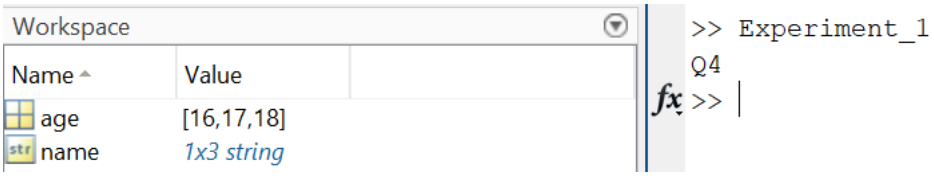


Q4) Create a string array containing the names of your three friends and their ages. Use writematrix command to write the string array to an excel file.

Code:

```
%4
disp("Q4");
name=["Alice" "Bob" "Chef"];
age=[16 17 18];
writematrix(name,"savexcel.xlsx");
writematrix(age,"savexcel.xlsx",'Writemode','append');
```

Output:

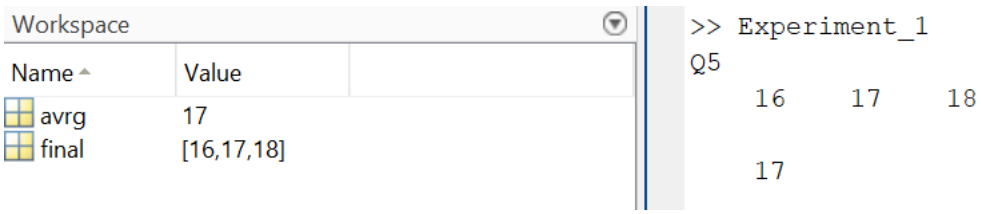


Q5) Use readmatrix command to read from the saved excel file and compute the average age of your three friends.

Code:

```
%5
disp("Q5");
final=readmatrix("savexcel.xlsx");
disp(final);
avrg=mean(final);
disp(avrg);
```

Output:

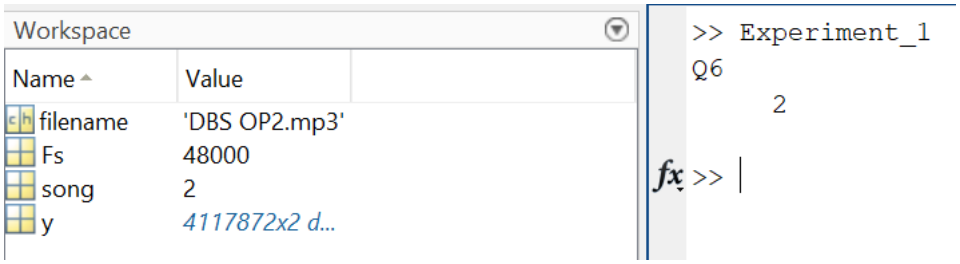


Q6) Use audioread command to read an audio file.

Code:

```
%6
disp("Q6");
song=randi([1,2],[1,1]);
disp(song);
filename='yes';
if(song==1)
    filename='DBS OP1.mp3';
else
    filename='DBS OP2.mp3';
end
[y,Fs] = audioread(filename);
sound(y,Fs);
```

Output:

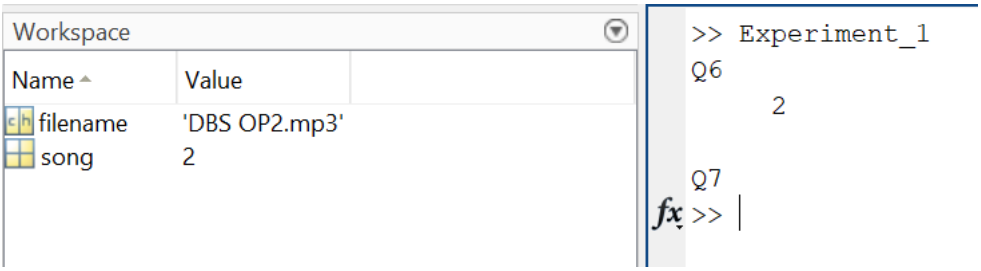


Q7) Use audiowrite command to write a portion of the file to a new audio file.

Code:

```
%7
disp("Q7");
audiowrite("DBnew.mp3",y,Fs);
clear y Fs;
```

Output:

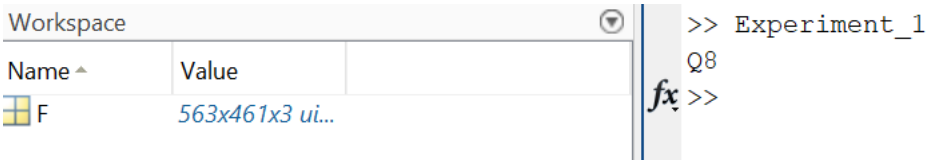


Q8) Use imread to read a jpg image to a three dimensional matrix F.

Code:

```
%8
disp("Q8");
F=imread("goku 00.jpg");
```

Output:

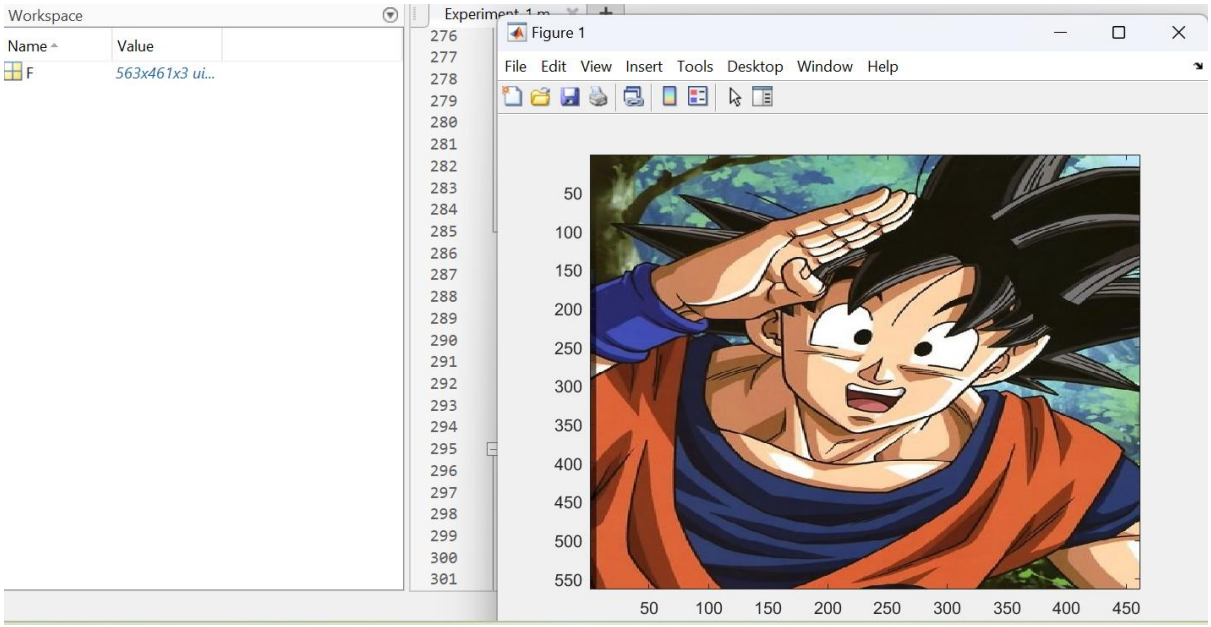


Q9) Use image function to display the image.

Code:

```
%9
disp("Q9");
image(F);
```

Output:

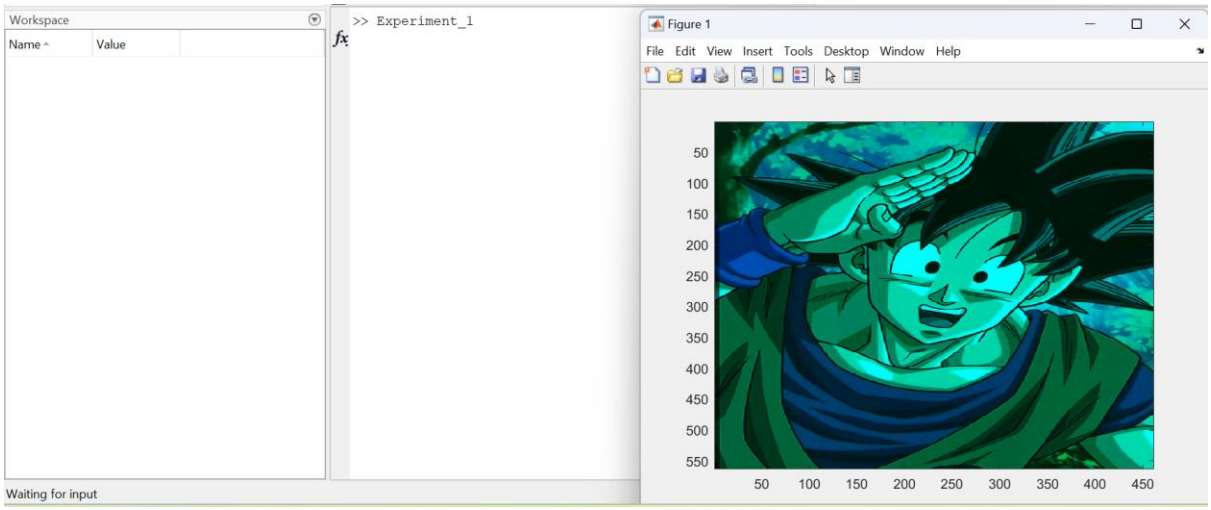


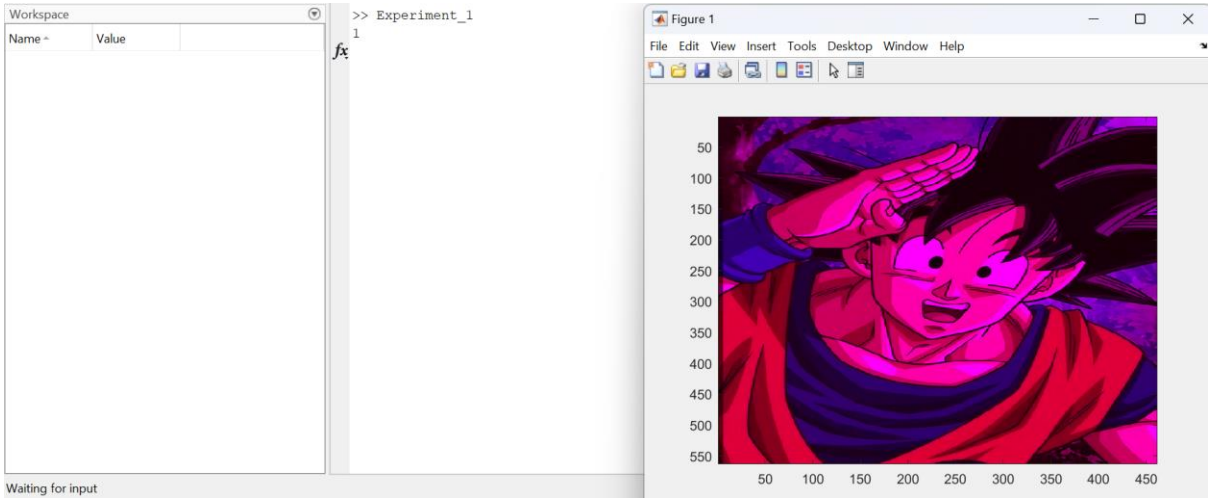
Q10) Make the entries in F(:, :, 1) all zero and display the image. Restore the image and do the same for F(:, :, 2) and F(:, :, 3).

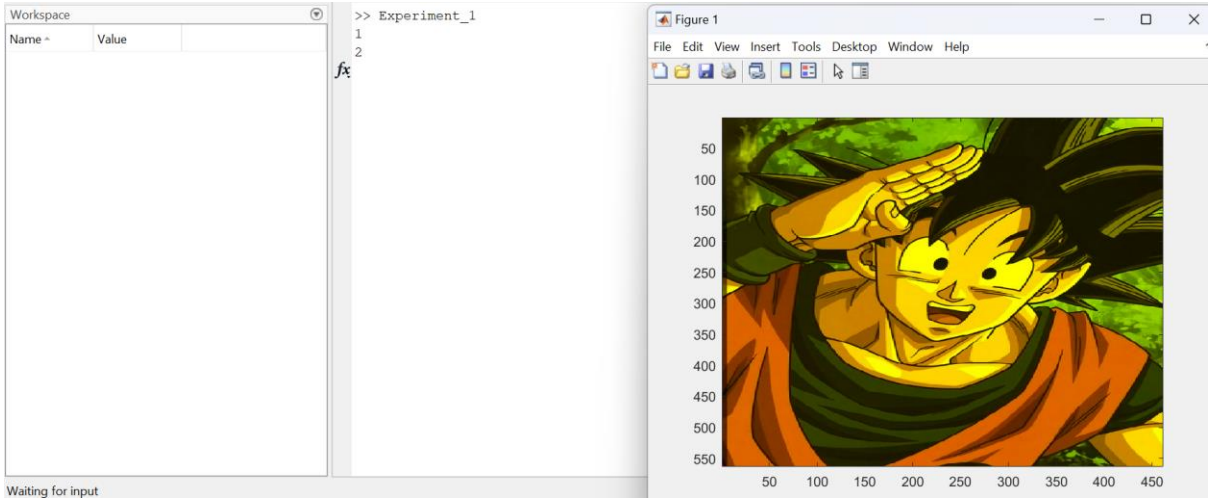
Code:

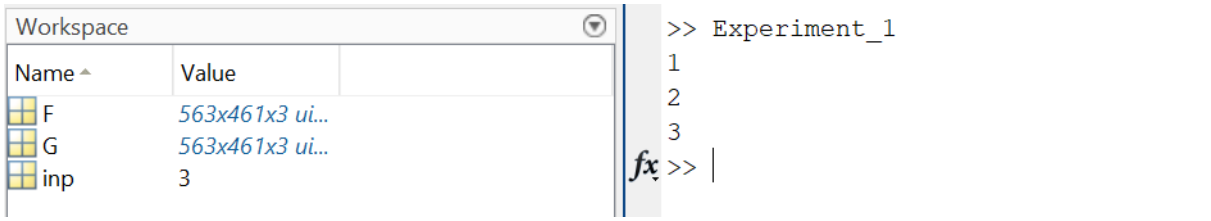
```
%10
disp("Q10");
G=F;
F(:,:,1)=0;
image(F);
inp=input("");
F=G;
F(:,:,2)=0;
image(F);
inp=input("");
F=G;
F(:,:,3)=0;
image(F);
inp=input("");
F=G;
```

Output:


Workspace
Name
Value
>> Experiment_1
Figure 1
File Edit View Insert Tools Desktop Window Help
Waiting for input


Workspace
Name
Value
>> Experiment_1
1
Figure 1
File Edit View Insert Tools Desktop Window Help
Waiting for input


Workspace
Name
Value
>> Experiment_1
1
2
Figure 1
File Edit View Insert Tools Desktop Window Help
Waiting for input


Workspace
Name
Value
F
563x461x3 ui...
G
563x461x3 ui...
inp
3
>> Experiment_1
1
2
3
>>

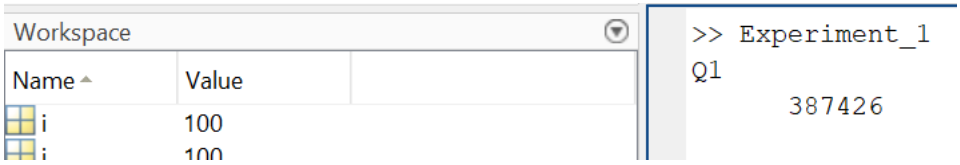
LOOPS AND FUNCTIONS

Q1) Create a 100 × 100 random matrix K having integer valued elements uniformly distributed in the interval 1 to 10. Use for loop to find the sum of square of the entries in the matrix K.

Code:

```
%1
disp("Q1");
K=randi([1,10],[100,100]);
sm=0;
for i=1:100
    for j=1:100
        sm=sm+K(i,j)*K(i,j);
    end
end
disp(sm);
```

Output:

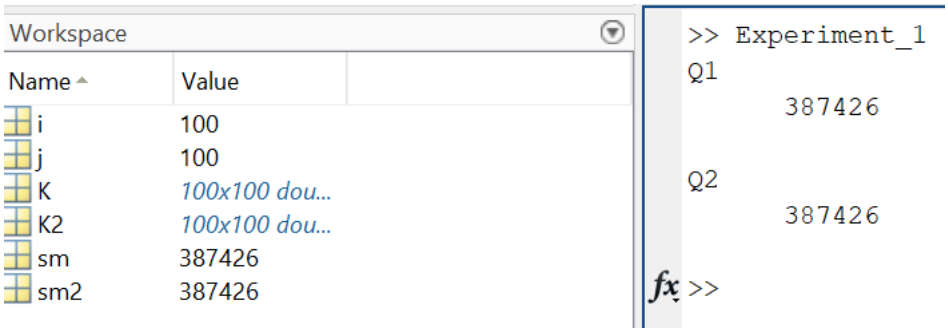


Q2) Do part 1 without using for loop.

Code:

```
%2
disp("Q2");
K2=K.*K;
sm2=sum(K2(:));
disp(sm2);
```

Output:

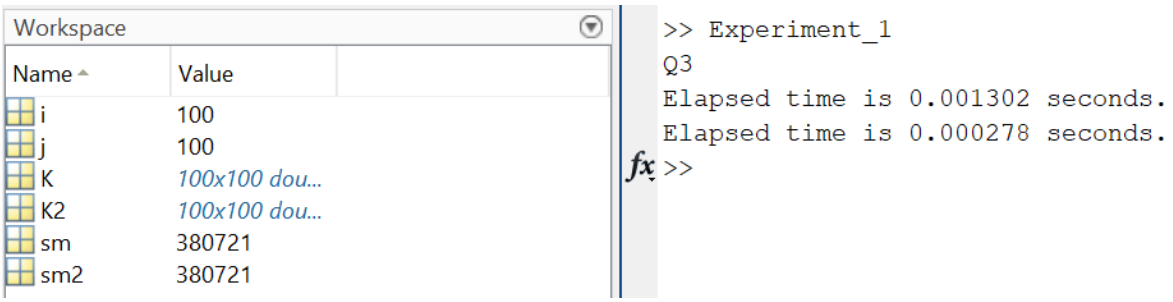


Q3) Use tic toc command to compute the time elapsed in Part 1 and Part 2. Read more about vectorization.

Code:

```
%3
disp("Q3");
tic
sm=0;
for i=1:100
    for j=1:100
        sm=sm+K(i,j)*K(i,j);
    end
end
toc
tic
K2=K.*K;
sm2=sum(K2(:));
toc
```

Output:

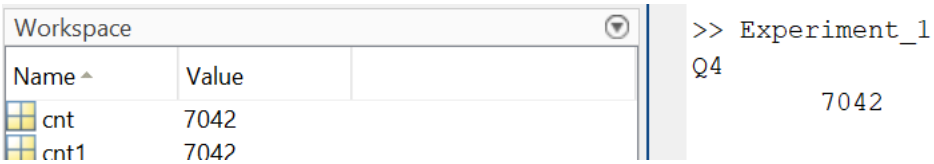


Q4) Find the number of entries greater than 5 and less than 3 in the generated matrix K. Use if else statement

Code:

```
%4
disp("Q4");
cnt=0;
for i=1:100
    for j=1:100
        if K(i,j)<3 || K(i,j)>5
            cnt=cnt+1;
        end
    end
end
disp(cnt);
```

Output:

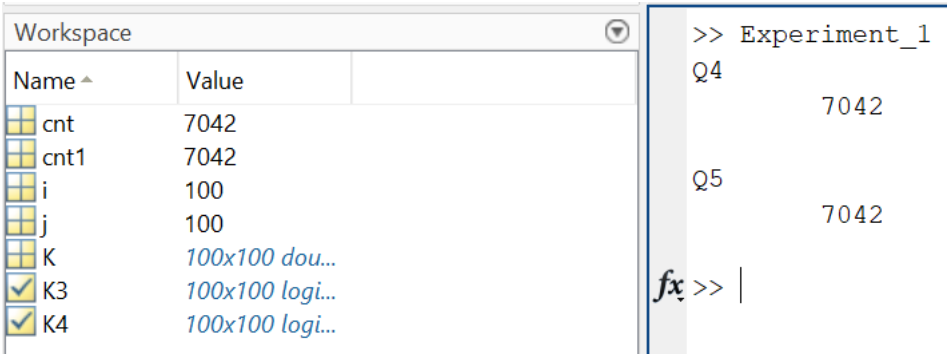


Q5) Do part 3 without using if else statement.

Code:

```
%5
disp("Q5");
K3=K>5;
K4=K<3;
cnt1=sum(K3(:))+sum(K4(:));
disp(cnt1);
```

Output:

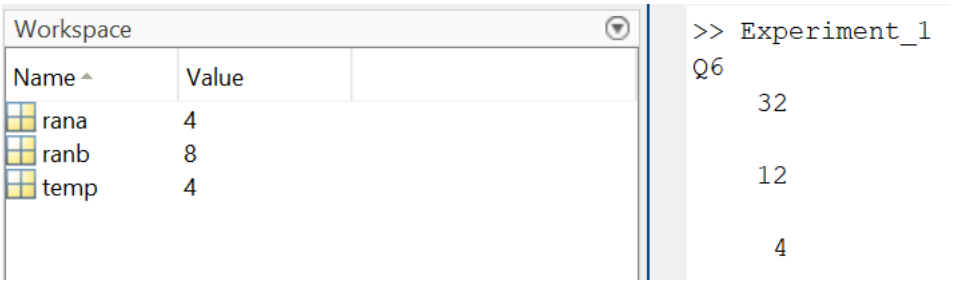


Q6) Demonstrate the use of while loop by a small program.

Code: (GCD/HCF of two integers a and b)

```
%6
disp("Q6");
ranb=randi([20,40]);
rana=randi([10,20]);
disp(ranb);
disp(rana);
while rem(ranb,rana)~=0
    temp=rem(ranb,rana);
    ranb=rana;
    rana=temp;

end
disp(rana);
```

Output:

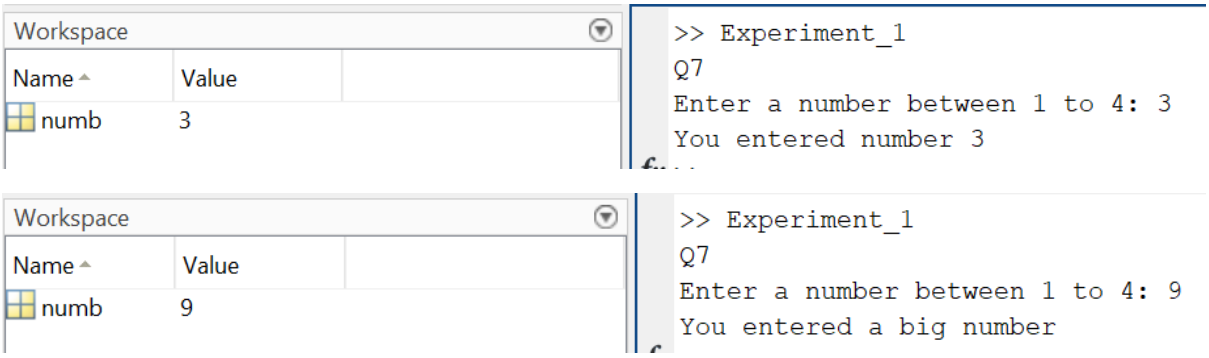


Q7) Demonstrate the use of switch statement by a small program.

Code:

```
%7
disp("Q7");
numb=input("Enter a number between 1 to 4: ");
switch(numb)
    case 1
        disp("You entered number 1");
    case 2
        disp("You entered number 2");
    case 3
        disp("You entered number 3");
    case 4
        disp("You entered numbe4 4");
    otherwise
        disp("You entered a big number");
end
```

Output:



Q8) Define a function to find the factorial of a number.

Code:

```
%8
disp("Q8");
n=input("Enter the number for your factorial: ");
fact=factorial(n);
disp(fact);
function fac=factorial(n)
    if(n==0) fac=1;
    else
        fac=n*(factorial(n-1));
    end
end
```

Output: