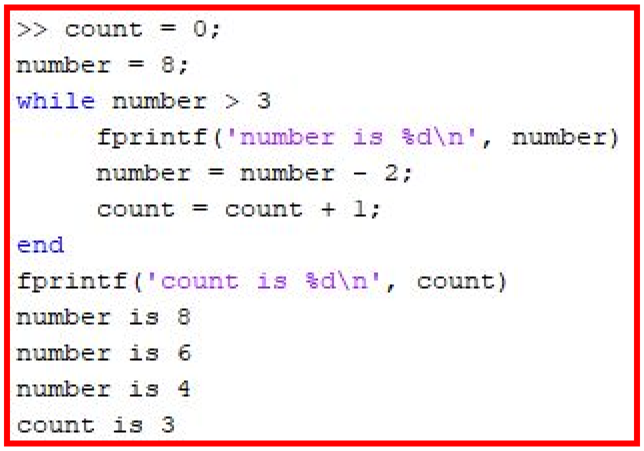
ENG EK 125 - Worksheet 5B

Name: Ande Chen   Section: C1

1) Trace this to figure out what the result will be, and then type it into MATLAB to verify the results.

```
count = 0;
number = 8;
while number > 3
     fprintf('number is %d\n', number)
     number = number - 2;
     count = count + 1;
end
fprintf('count is %d\n', count)
```

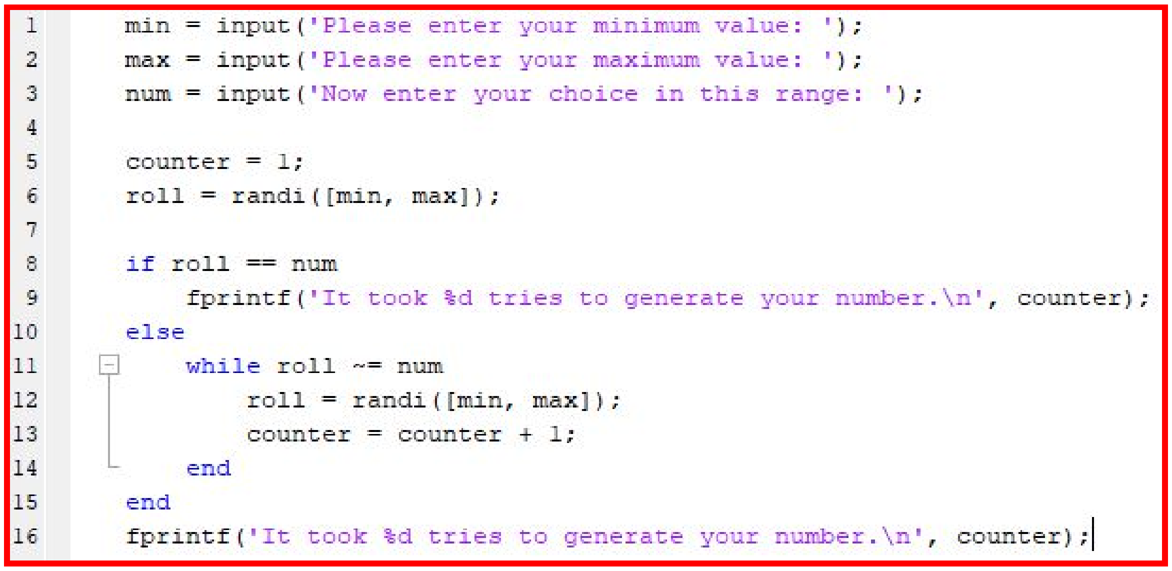
count = 3



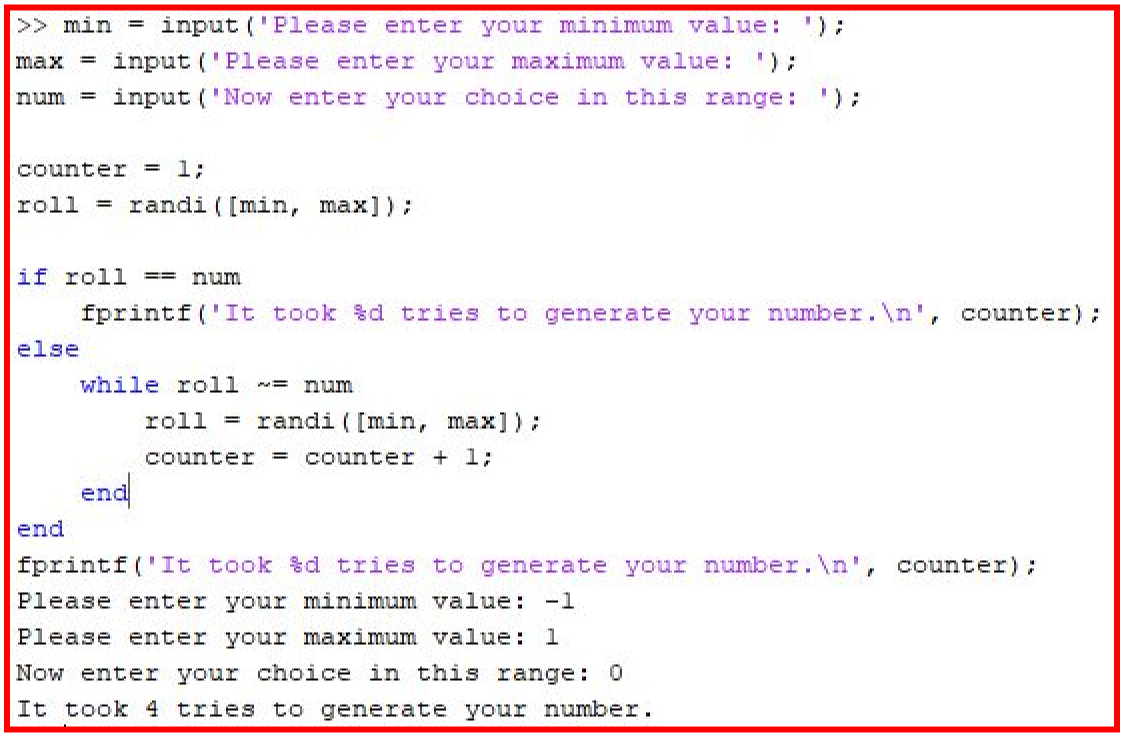
```
>> count = 0;
number = 8;
while number > 3
     fprintf('number is %d\n', number)
     number = number - 2;
     count = count + 1;
end
fprintf('count is %d\n', count)
number is 8
number is 6
number is 4
count is 3
```

2) Write a script (for example, called *findmine*) that will prompt the user for minimum and maximum integers, and then another integer which is the user's choice in the range from the minimum to the maximum. The script will then generate random integers in the range from the minimum to the maximum, until a match for the user's choice is generated. The script will print how many random integers had to be generated until a match for the user's choice was found. For example, running this script might result in this output:

```
>> findmine
Please enter your minimum value: -2
Please enter your maximum value: 3
Now enter your choice in this range: 0
It took 3 tries to generate your number
```

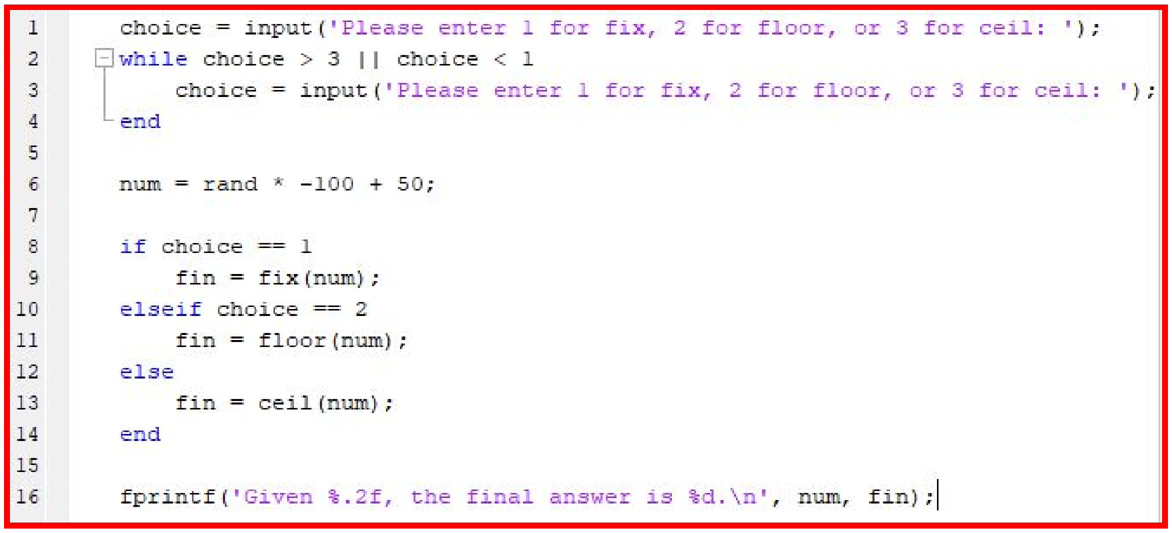


```
min = input('Please enter your minimum value: ');
max = input('Please enter your maximum value: ');
num = input('Now enter your choice in this range: ');

counter = 1;
roll = randi([min, max]);

if roll == num
    fprintf('It took %d tries to generate your number.\n', counter);
else
    while roll ~= num
        roll = randi([min, max]);
        counter = counter + 1;
    end
end
fprintf('It took %d tries to generate your number.\n', counter);
```

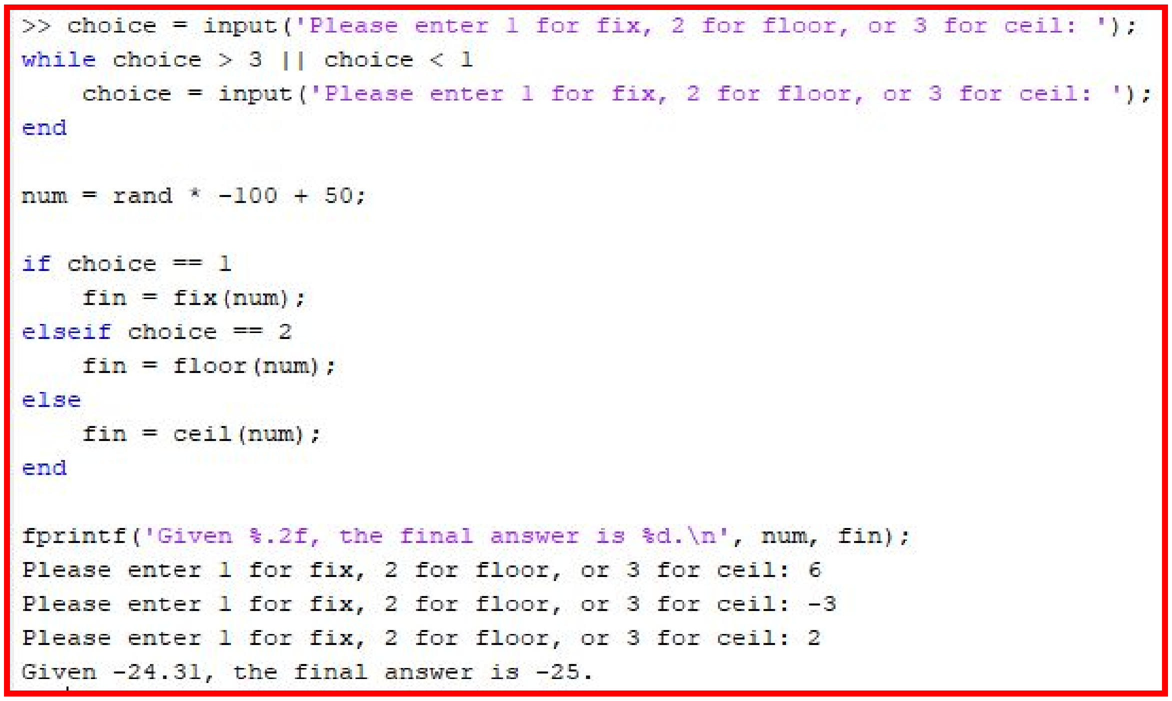


```
>> min = input('Please enter your minimum value: ');
max = input('Please enter your maximum value: ');
num = input('Now enter your choice in this range: ');

counter = 1;
roll = randi([min, max]);

if roll == num
    fprintf('It took %d tries to generate your number.\n', counter);
else
    while roll ~= num
        roll = randi([min, max]);
        counter = counter + 1;
    end
end
fprintf('It took %d tries to generate your number.\n', counter);
Please enter your minimum value: -1
Please enter your maximum value: 1
Now enter your choice in this range: 0
It took 4 tries to generate your number.
```

3) Write a script that will present the user with choices for functions “1 for fix”, “2 for floor”, and “3 for ceil”. Error-check by looping to display the menu until the user enters a valid choice. Then, generate a random number and print the result of the user’s function choice of that number (e.g. **fix(5)**). You may just print the menu, or use the **menu** function, or the **listdlg** function.



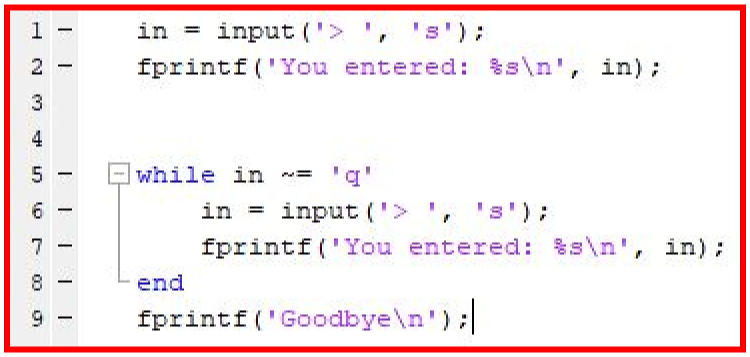
```
choice = input('Please enter 1 for fix, 2 for floor, or 3 for ceil: ');
while choice > 3 || choice < 1
    choice = input('Please enter 1 for fix, 2 for floor, or 3 for ceil: ');
end

num = rand * -100 + 50;

if choice == 1
    fin = fix(num);
elseif choice == 2
    fin = floor(num);
else
    fin = ceil(num);
end

fprintf('Given %.2f, the final answer is %d.\n', num, fin);
```



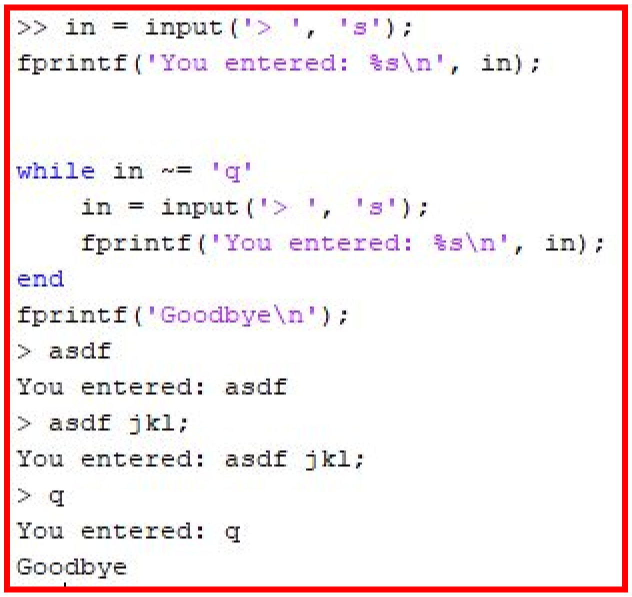
```
>> choice = input('Please enter 1 for fix, 2 for floor, or 3 for ceil: ');
while choice > 3 || choice < 1
    choice = input('Please enter 1 for fix, 2 for floor, or 3 for ceil: ');
end

num = rand * -100 + 50;

if choice == 1
    fin = fix(num);
elseif choice == 2
    fin = floor(num);
else
    fin = ceil(num);
end

fprintf('Given %.2f, the final answer is %d.\n', num, fin);
Please enter 1 for fix, 2 for floor, or 3 for ceil: 6
Please enter 1 for fix, 2 for floor, or 3 for ceil: -3
Please enter 1 for fix, 2 for floor, or 3 for ceil: 2
Given -24.31, the final answer is -25.
```

4) Unix is a command line driven operating system. The Unix command line behaves similar to the MATLAB interpreter: users repeatedly enter commands at a prompt (denoted by the character '>'). You are to write a "command line" that repeatedly prompts the user for a command and repeats it back to them. The user can exit the command line when only the character 'q' is passed (meaning "quit"). For example:

```
>> command_line
> ls ../folder
You entered: ls ../folder
> echo hello world!
You entered: echo hello world!
> q
Goodbye
```



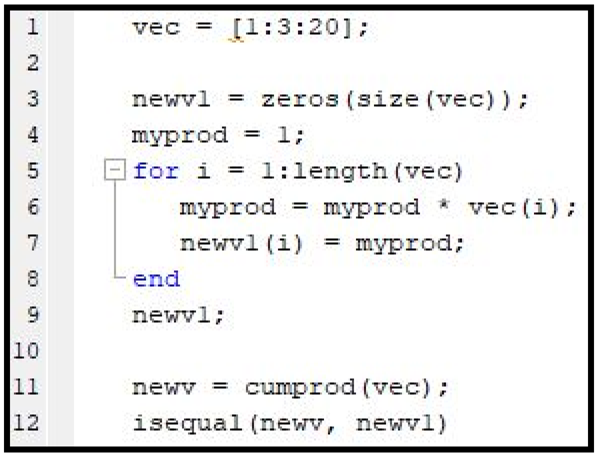
```
in = input('> ', 's');
fprintf('You entered: %s\n', in);


while in ~= 'q'
    in = input('> ', 's');
    fprintf('You entered: %s\n', in);
end
fprintf('Goodbye\n');
```



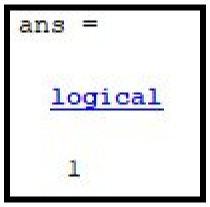
```
>> in = input('> ', 's');
fprintf('You entered: %s\n', in);


while in ~= 'q'
    in = input('> ', 's');
    fprintf('You entered: %s\n', in);
end
fprintf('Goodbye\n');
> asdf
You entered: asdf
> asdf jkl;
You entered: asdf jkl;
> q
You entered: q
Goodbye
```

5) Vectorize this code! Write *one* assignment statement that will accomplish exactly the same thing as the given code (assume that the variable *vec* has been initialized):

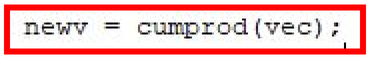
```
newv = zeros(size(vec));
myprod = 1;
for i = 1:length(vec)
   myprod = myprod * vec(i);
   newv(i) = myprod;
end
newv % Note: this is just to display the value
```



```
vec = [1:3:20];

newv1 = zeros(size(vec));
myprod = 1;
for i = 1:length(vec)
    myprod = myprod * vec(i);
    newv1(i) = myprod;
end
newv1;

newv = cumprod(vec);
isequal(newv, newv1)
```

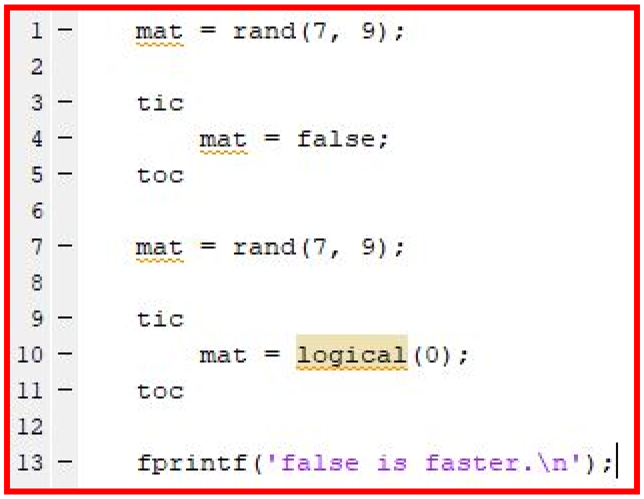


```
ans =

  logical

   1
```

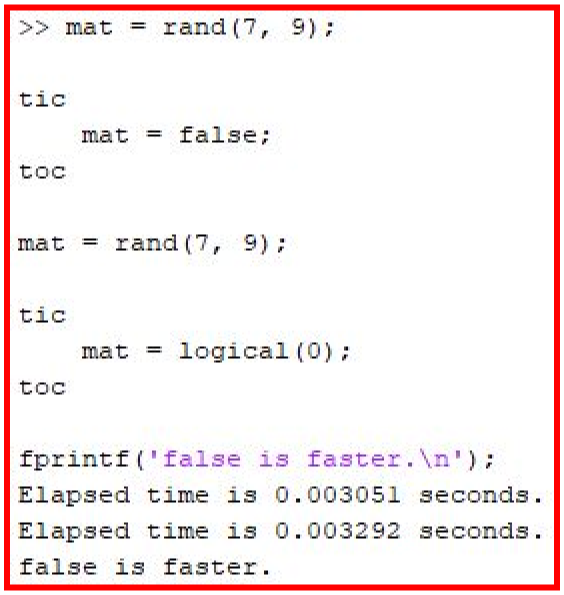


```
newv = cumprod(vec);
```

6) Which is faster: using **false** or using **logical**(0) to preallocate a matrix to all **logical** zeros? Write a script to test this.



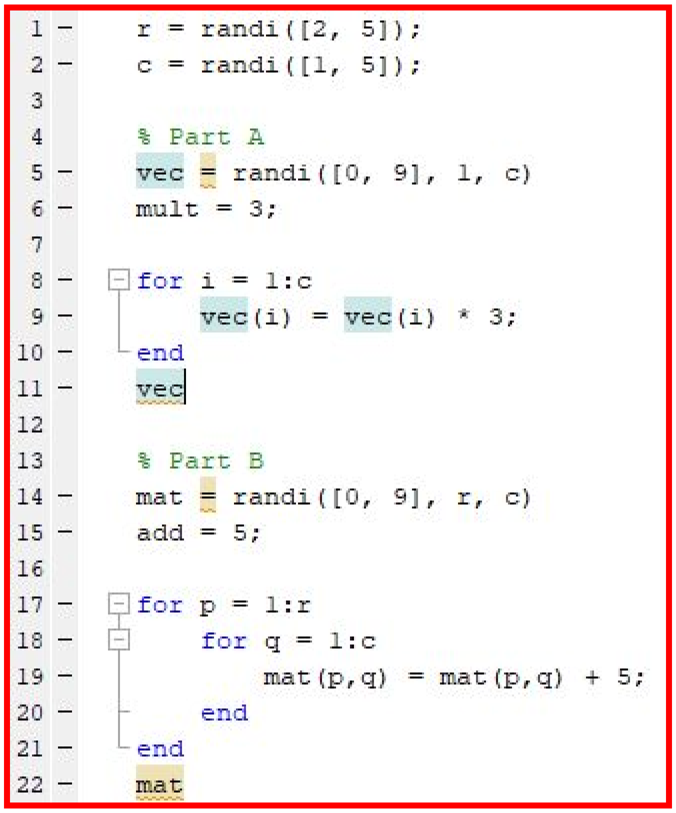
```
mat = rand(7, 9);

tic
    mat = false;
toc

mat = rand(7, 9);

tic
    mat = logical(0);
toc

fprintf('false is faster.\n');
```



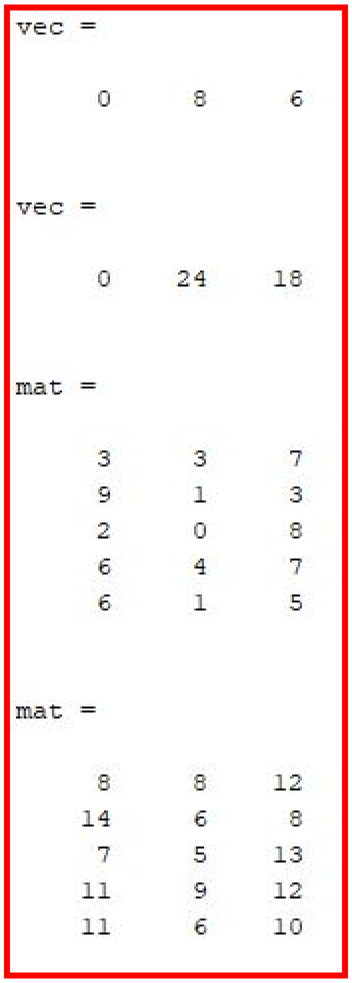
```
>> mat = rand(7, 9);

tic
    mat = false;
toc

mat = rand(7, 9);

tic
    mat = logical(0);
toc

fprintf('false is faster.\n');
Elapsed time is 0.003051 seconds.
Elapsed time is 0.003292 seconds.
false is faster.
```

7) And now, for the reverse of vectorizing. Given a vector variable *vec* and a matrix variable *mat*, write code using for loops, if statements, etc. that will accomplish the same as each of the following:

```
a)  vec * 3
b)  mat + 5
```



```
r = randi([2, 5]);
c = randi([1, 5]);

% Part A
vec = randi([0, 9], 1, c)
mult = 3;

for i = 1:c
    vec(i) = vec(i) * 3;
end
vec

% Part B
mat = randi([0, 9], r, c)
add = 5;

for p = 1:r
    for q = 1:c
        mat(p,q) = mat(p,q) + 5;
    end
end
mat
```



```
vec =

     0     8     6


vec =

     0    24    18


mat =

     3     3     7
     9     1     3
     2     0     8
     6     4     7
     6     1     5


mat =

     8     8    12
    14     6     8
     7     5    13
    11     9    12
    11     6    10
```