SVKM'S NMIMS Deemed to be UNIVERSITY | NAVI MUMBAI

MATLAB
Unit 6-Lecture 24
“while ... end" loop

BTech (CSBS) -Semester VII

7 October 2022, 09:35AM

Control Flow and Operators

1) relational and logical operators
2) “if ... end" structure
3) “for ... end" loop
4) “while ... end" loop
5) other flow structures
6) operator precedence
7) saving output to a file

“while” loop

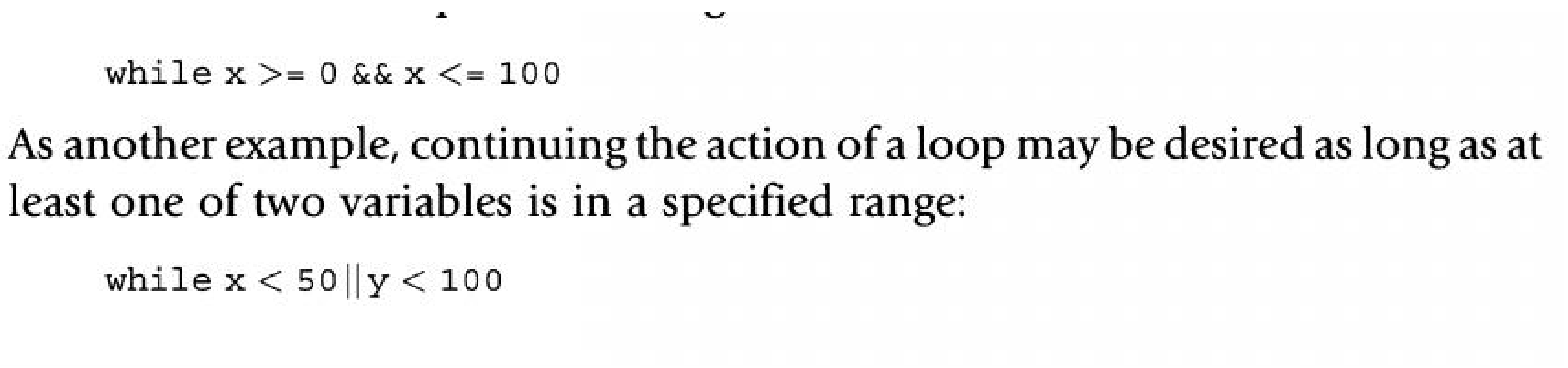
```
while condition
    action
end
```

factgthigh.m

```
function facgt = factgthigh(high)
% factgthigh returns the first factorial > input
% Format: factgthigh(inputInteger)

i=0;
fac=1;
while fac <= high
    i=i+1;
    fac = fac * i;
end
facgt = fac;
end
```

```
>> factgthigh(5000)
ans =
        5040
```

Multiple condition in "while" loop



```
while x >= 0 && x <= 100
```

As another example, continuing the action of a loop may be desired as long as at least one of two variables is in a specified range:

```
while x < 50||y < 100
```

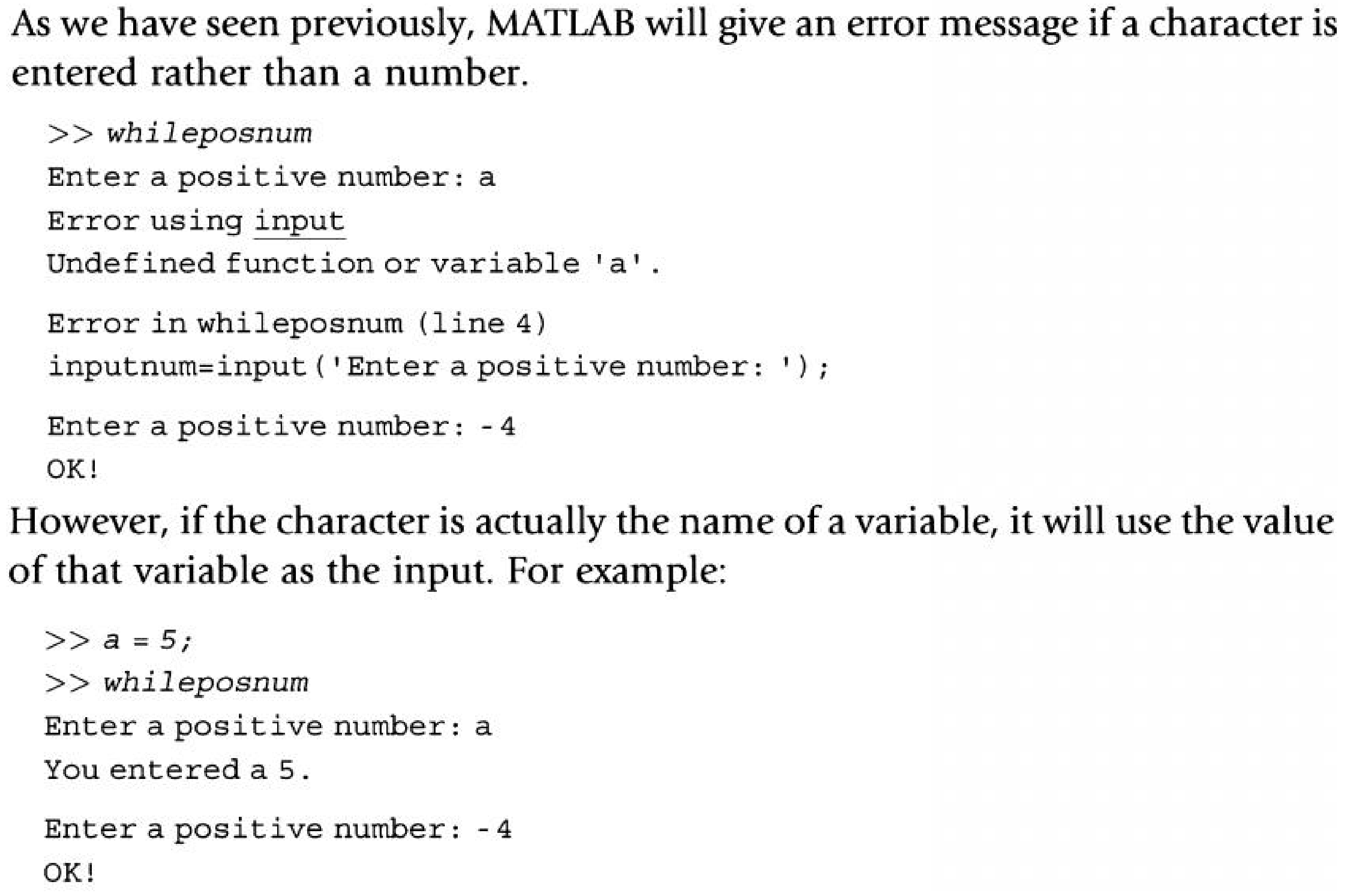
Input in “while” loop

whileposnum.m

```
% Prompts the user and echo prints the numbers entered
% until the user enters a negative number

inputnum=input('Enter a positive number: ');
while inputnum >= 0
    fprintf('You entered a %d.\n\n',inputnum)
    inputnum = input('Enter a positive number: ');
end
fprintf('OK!\n')
```

```
>> whileposnum
Enter a positive number: 6
You entered a 6.

Enter a positive number: -2
OK!
```

Input in “while” loop



As we have seen previously, MATLAB will give an error message if a character is entered rather than a number.

```
>> whileposnum
Enter a positive number: a
Error using input
Undefined function or variable 'a'.

Error in whileposnum (line 4)
inputnum=input('Enter a positive number: ');

Enter a positive number: -4
OK!
```

However, if the character is actually the name of a variable, it will use the value of that variable as the input. For example:

```
>> a = 5;
>> whileposnum
Enter a positive number: a
You entered a 5.

Enter a positive number: -4
OK!
```

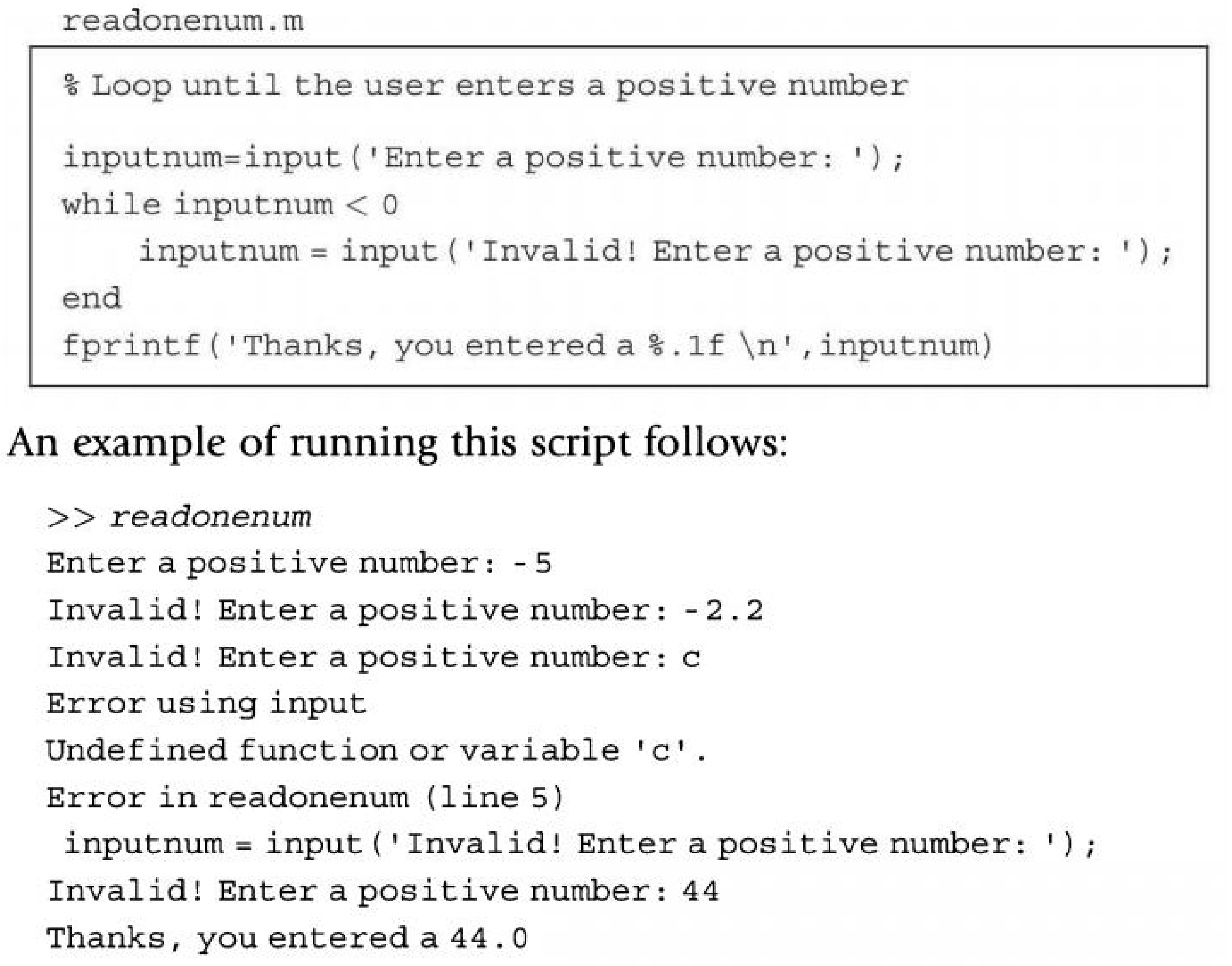
Counting in “while” loop

countposnum.m

```
% Prompts the user for positive numbers and echo prints as
% long as the user enters positive numbers

% Counts the positive numbers entered by the user
counter=0;
inputnum=input('Enter a positive number: ');
while inputnum >= 0
     fprintf('You entered a %d.\n\n',inputnum)
     counter = counter+1;
     inputnum = input('Enter a positive number: ');
end
fprintf('Thanks, you entered %d positive numbers.\n',counter)
```

```
>> countposnum
Enter a positive number: 4
You entered a 4.

Enter a positive number: 11
You entered a 11.

Enter a positive number: -4
Thanks, you entered 2 positive numbers.
```

Error checking input in “while” loop



readonenum.m

```
% Loop until the user enters a positive number

inputnum=input('Enter a positive number: ');
while inputnum < 0
    inputnum = input('Invalid! Enter a positive number: ');
end
fprintf('Thanks, you entered a %.1f \n',inputnum)
```

An example of running this script follows:

```
>> readonenum
Enter a positive number: -5
Invalid! Enter a positive number: -2.2
Invalid! Enter a positive number: c
Error using input
Undefined function or variable 'c'.
Error in readonenum (line 5)
 inputnum = input('Invalid! Enter a positive number: ');
Invalid! Enter a positive number: 44
Thanks, you entered a 44.0
```

Question

How could we vary the previous example so that the script asks the user to enter positive numbers *n* times, where *n* is an integer defined to be 3?

Answer: Every time the user enters a value, the script checks and in a **while** loop keeps telling the user that it's invalid until a valid positive number is entered. By putting the error-check in a **for** loop that repeats n times, the user is forced eventually to enter three positive numbers, as shown in the following.

readnnums.m

```
% Loop until the user enters n positive numbers
n=3;
fprintf('Please enter %d positive numbers\n\n',n)
for i=1:n
    inputnum=input('Enter a positive number: ');
    while inputnum < 0
        inputnum = input('Invalid! Enter a positive number: ');
    end
    fprintf('Thanks, you entered a %.1f \n',inputnum)
end
```

```
>> readnnums
Please enter 3 positive numbers

Enter a positive number: 5.2
Thanks, you entered a 5.2
Enter a positive number: 6
Thanks, you entered a 6.0
Enter a positive number: -7.7
Invalid! Enter a positive number: 5
Thanks, you entered a 5.0
```

Question

MATLAB has a **cumsum** function that will return a vector of all of the running sums of an input vector. However, many other languages do not, so how could we write our own?

Answer: Essentially, there are two programming methods that could be used to simulate the **cumsum** function. One method is to start with an empty vector and extend the vector by adding each running sum to it as the running sums are calculated. A better method is to preallocate the vector to the correct size and then change the value of each element to be successive running sums.

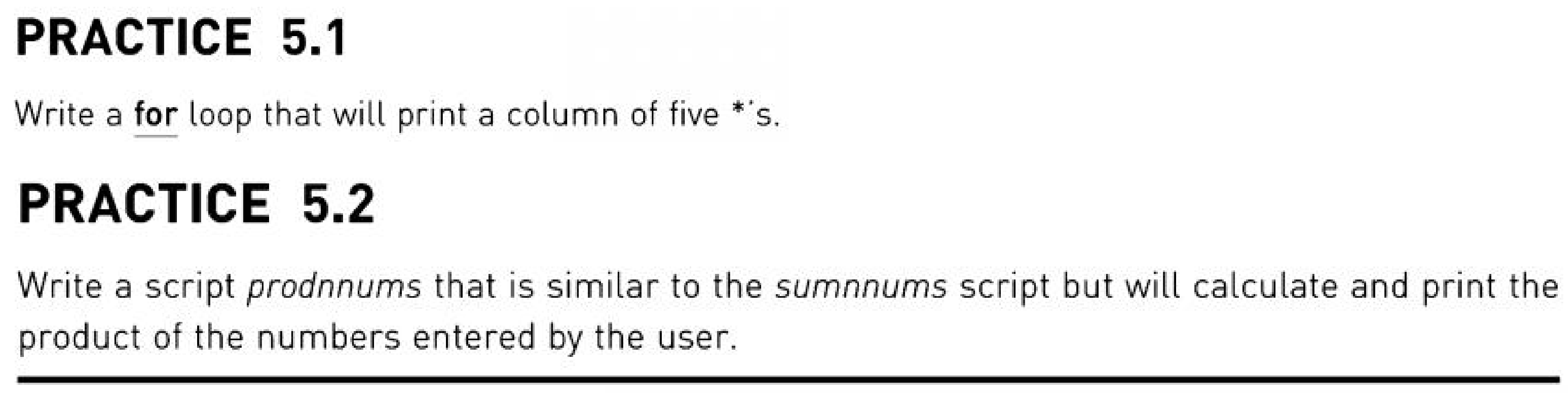
myveccumsum.m

```
function outvec = myveccumsum(vec)
% myveccumsum imitates cumsum for a vector
% It preallocates the output vector
% Format: myveccumsum(vector)

outvec = zeros(size(vec));
runsum = 0;
for i = 1:length(vec)
    runsum = runsum + vec(i);
    outvec(i) = runsum;
end
end
```

An example of calling the function follows:

```
>> myveccumsum([5   9   4])
ans =
     5   14   18
```

Practise question



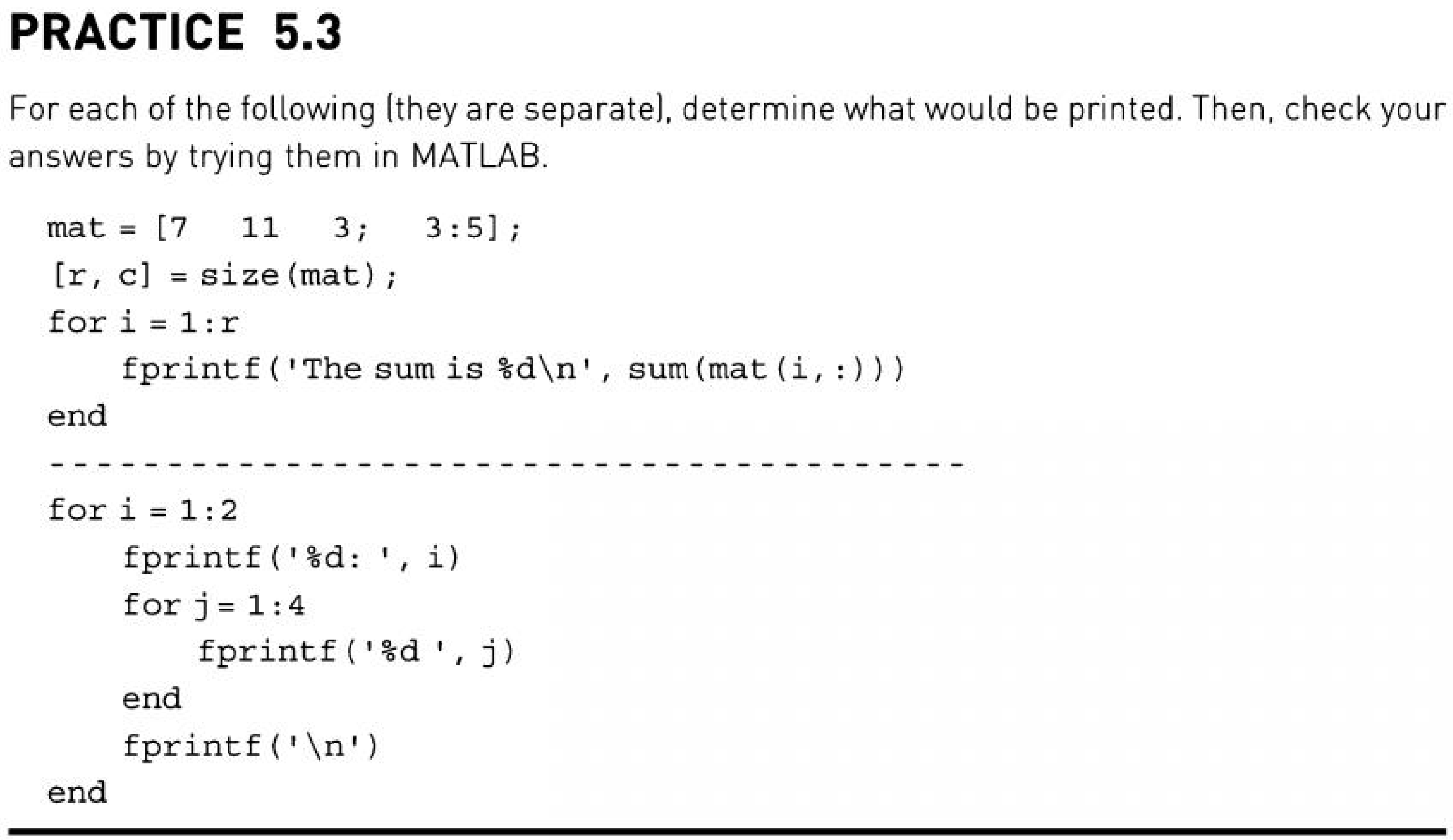
PRACTICE 5.1

Write a **for** loop that will print a column of five *'s.

PRACTICE 5.2

Write a script *prodnnums* that is similar to the *sumnnums* script but will calculate and print the product of the numbers entered by the user.

Practise question

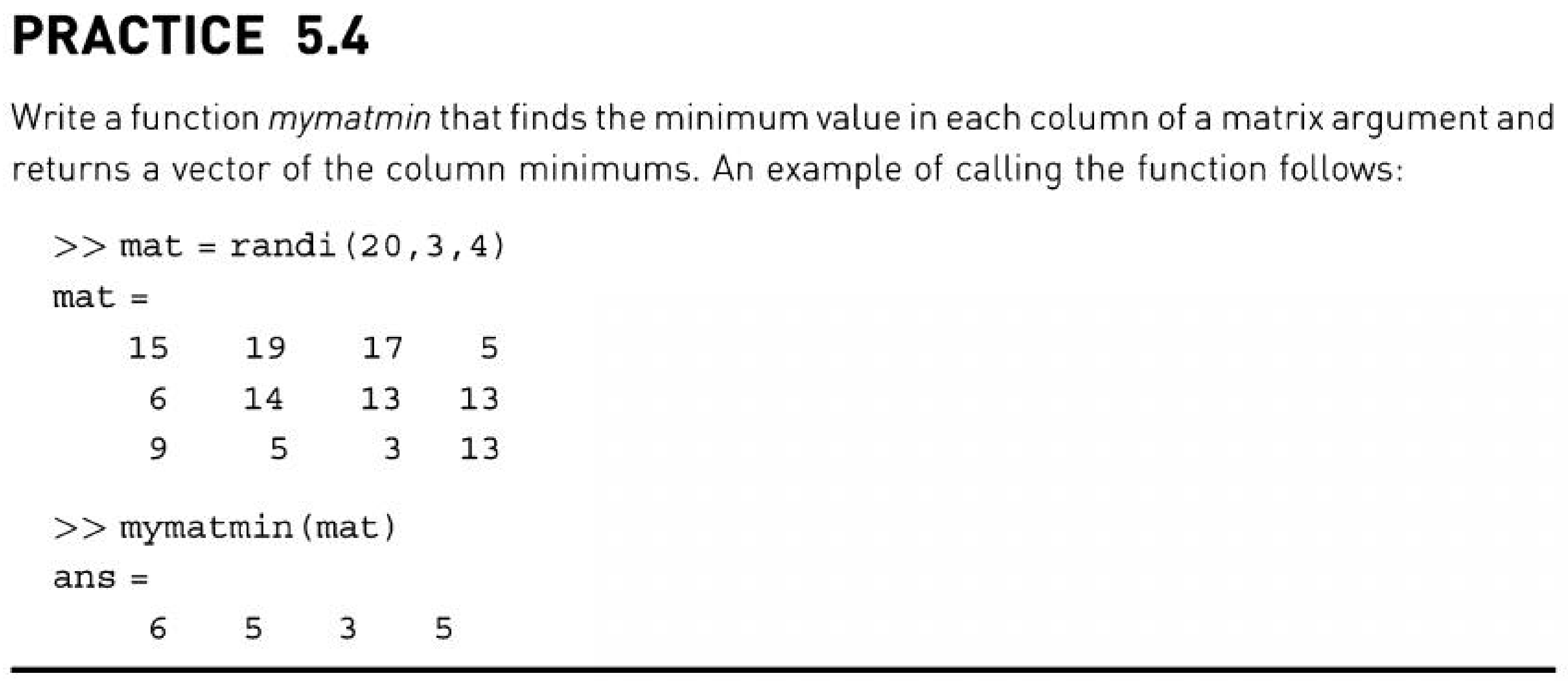


PRACTICE 5.3

For each of the following (they are separate), determine what would be printed. Then, check your answers by trying them in MATLAB.

```
mat = [7   11   3;   3:5];
[r, c] = size(mat);
for i = 1:r
    fprintf('The sum is %d\n', sum(mat(i,:)))
end
----------------------------------------
for i = 1:2
    fprintf('%d: ', i)
    for j = 1:4
        fprintf('%d ', j)
    end
    fprintf('\n')
end
```

Practise question

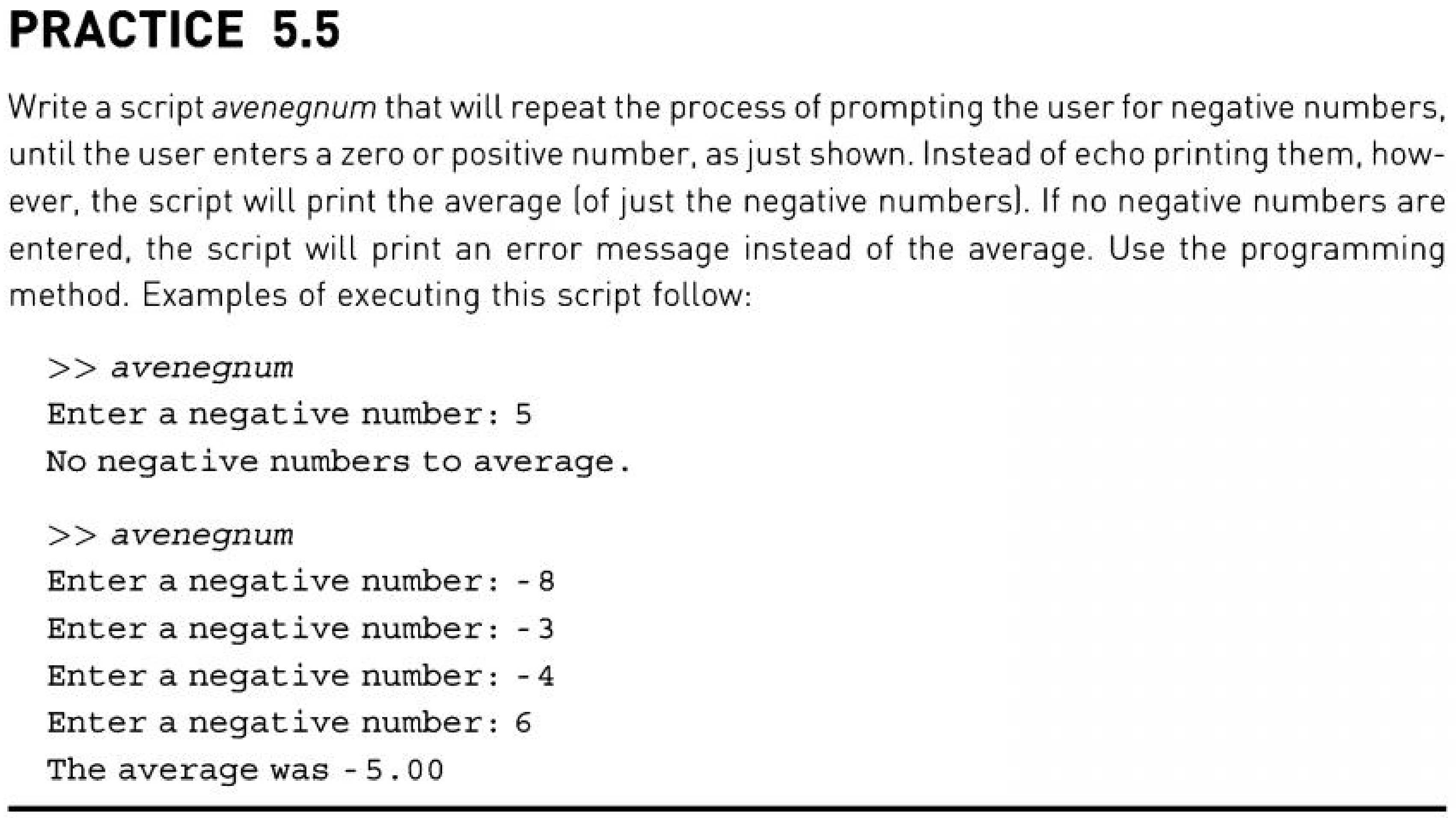


PRACTICE 5.4

Write a function *mymatmin* that finds the minimum value in each column of a matrix argument and returns a vector of the column minimums. An example of calling the function follows:

```
>> mat = randi(20,3,4)
mat =
    15    19    17     5
     6    14    13    13
     9     5     3    13

>> mymatmin(mat)
ans =
     6     5     3     5
```

Practise question



PRACTICE 5.5

Write a script *avenegnum* that will repeat the process of prompting the user for negative numbers, until the user enters a zero or positive number, as just shown. Instead of echo printing them, however, the script will print the average (of just the negative numbers). If no negative numbers are entered, the script will print an error message instead of the average. Use the programming method. Examples of executing this script follow:

```
>> avenegnum
Enter a negative number: 5
No negative numbers to average.

>> avenegnum
Enter a negative number: -8
Enter a negative number: -3
Enter a negative number: -4
Enter a negative number: 6
The average was -5.00
```