Problem 1.
Let the factorial function $n!$, for any $n \geq 1$, be defined as the product $n \cdot (n-1) \cdot (n-2) \cdot 2 \cdot 1$. That is, $3! = 3 \cdot 2 \cdot 1 = 6$. Write MATLAB code that computes $n!$ for a variable n already defined to have a value that is a positive integer. Compute the result (factorial) and store it in variable z.

Problem 2.
Compute the sum of all the elements of matrix a that is an $m \times n$ matrix. You may use the sum function but only once (i.e. `sum(sum(a))`) is not a solution!

Problem 3.
(a) Let `FourA` be a row vector. Write a correct piece of MATLAB code that counts the number of 1s in that vector. For example if vector `FourA` is [1 2 1 100 1.0] the answer will be 3.
(b) Let `FourB` be a row vector. Write a correct piece of MATLAB code that changes all 1s into 5s. For example if `FourB` is [1 2 1 100 1.0] the answer will be [5 2 5 100 5] afterwards.

Problem 4.
In a 5×5 matrix $a = round(rand(5,5) * 10)$, delete its 3-rd row and then its 3-rd column.

Problem 5.
What does the MATLAB statement(s) do?

```
>> a= zeros(10,10);
>> a(:) = 1:100;
```

Problem 6.
You are given two n-element row vectors x, y which are the coordinates $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ of two points in n dimensional Euclidean space. You are asked to compute the norm

$$||x - y|| = \sqrt{|x_1 - y_1|^2 + \ldots + |x_n - y_n|^2}.$$

How do you do it in MATLAB? (Note you know x,y have the same number of elements, but you still don't know n.)

Problem 7.
The n-th approximation of e is e_n where $e_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{n!}$. Compute e_n using array operations!

Turn page for solutions

Solutions

Problem 1.

```
v1 =10:-1:1;
v2 = v1([10:-3 1 2:2:8]);
v1(2:4) = 10:-5:0;
v3 =[ 1:3 1:3 1:3];

v1(6:8) = 10:10:30;
```

Problem 2.
The code is simple; we generate a row vector with the values $1, 2, \ldots, n$ and then multiply them together.

```
% n is already defined i.e. assigned a non-negative integer value!
% say n=5
vector = 1:n;               % the first two lines could have also become
z = prod(vector);           % z = prod(1:n);
```

The two lines can be combined into one. The statement `z= prod(1:n)` could do the same in one line! This is an **inline** solution! An alternative is to do `z=cumprod(1:n)`, and then retrieve `b(n)`.

Problem 3.

```
>>  sum(a(:))   % The colon : operator used for array indexing
                % 'converts' mxn matrix a into
                % a column vector of mn elements.
                % Thus one application of sum suffices!
```

Problem 4.
(a) The code first uses the relational operator `==` as an array operator to determine the elements of `fourA` that are equal to 1. Thus a row vector `z` is generated with entries 0 or 1 for false and true depending on whether the corresponding entry of `fourA` did not contain a 1 or it did hold a 1. The `sum` function applied to `z` sums up its elements (i.e. each 1 contributes 1 to the sum). And the `count` is the number of 1s in `z` i.e. the number of 1s in `fourA`; it is displayed in the third line of the code.

```
z= fourA == 1;
count=sum(z);
disp(count);
```

(b) We continue part (a). Variable `z` serves the previous intent of an indicator variable that shows (through an 1 entry) where an 1 appears in `FourB`. To replace those 1s in `FourB` with a 5 it suffices to add $(5-1) = 4$ to each such one and 0 everywhere else. Thus we can turn the 1s in z into 4s, and retain the zeroes, and then add the new values i.e. `(5-1)*z` into `fourB`.

```
z= fourB == 1;
fourB = fourB+ (5-1)*z;
```

Problem 5.

```
>> a(3,:)=[];
>> a(:,3)=[];
```

Problem 6.

The first statement generates a 10×10 matrix a of zeroes. The statement $1 : 100$ generates a row vector with the 100 integers from 1 to 100. The operation `a(:) = 1:100` fills a as a column vector because it uses array indexing. Thus the first column of a becomes $1, \ldots, 10$ and so on.

Problem 7.

We learn a new operation in this problem.

```
>> norm(x-y)                    % simple way

>> sqrt(sum(abs(x-y) .^2 ));   % array operations way!
```

Problem 8.

```
>>  n = input(' Provide a value for n ');
>>  a = 1: n    ;      % generate  1   2  3  ... n
>>  a = cumprod(a);    % generate  1!  2! e! ... n!
>>  a = 1 ./ a    ;    % generate 1/1! 1/2!  ... 1/n!
>>  a = sum(a)    ;    % add up
>>  a = a+1       ;    % add one for 1/0!
>>  disp(a)       ;    % this is e_n the n-th approximation of e
```