Introduction to Scientific Computing, Homework #2

Problem 1 Solution

```
clear;clc;
P = @(r, A, n, k)(r.*A.*((1+r./n).^(n.*k))./(n.*((1+r./n).^(n.*k)-1)));
% Define an anonymous function as a short-cut

% Method1: nested 'for' loops
% MATLAB is column-major, hence the outer loop is for row, the inner loop
% is for column. This leads to potential performance boosts, like enabling
% cache prefetching or instruction-level parallelism.
col = 1;
table1 = zeros(11, 3);
for year=15:5:25
    row = 1;
    for rate=0.1:0.01:0.2
        table1(row, col) = P(rate, 1000, 12, year);
        row = row + 1;
    end
    col = col + 1;
end
disp(table1);

% Method2: vectorized outer loop(maybe method3 is expected answer)
table2 = zeros(11, 3);
row = 1;
for rate=0.1:0.01:0.2
    table2(row,:) = P(rate, 1000, 12, 15:5:25);
    row = row + 1;
end
disp(table2);

% Method3(Extra): fully vectorized
% !NOTE: actually I don't really know what "outer loop" refers to
% so I vectorized ALL loops(inner & outer), that will definitely meet every
% requirements, right?
table3 = P((0.1:0.01:0.2)', 1000, 12, 15:5:25);
disp(table3);
```

Output Omitted. Exactly the same as the table in the question(but tripled).

```
>>10.7461    9.6502    9.0870

   11.3660   10.3219    9.8011

   12.0017   11.0109   10.5322

(…)
```

(Next Page)

Problem 2 Solution

```
% Hah, classic coding practice.
clear;clc;
disp(genNarcissistic());
function [ nars ] = genNarcissistic()
% Find all the 3-digits numbers which is "Narcissistic"

nars = [];
for i=100:999 % Enumerate all the 3-digits num
    if isNarcissistic(i)
        nars(end+1) = i;
        % The warning here indicates that a vector with variable length may lead
        % to performance loss. But since the size of this specific problem is
        % extremely small, we can just ignore it.
    end
end
end

function ret = isNarcissistic(num)
% Figure out whether the given num is a Narcissistic number or not

a = floor(num/100); % Extract the first digit
b = floor(rem(num, 100)/10); % Extract the second digit
c = rem(num, 10); % Extract the last digit
ret = (num == a^3 + b^3 + c^3); % Return in logical type
end
```

Output:

```
>>    153   370   371   407
```

(Next Page)

Problem 3 Solution

```
clear;clc;
A = [
    1,3,2;
    8,4,6;
    7,9,5]; % Matrix that is given

B = A(1:2:3 , 1:2); % Generate a new matrix from specific row/column
C = reshape(A(1:4), [2,2])';% Unwounded index, then be resized and transposed
D = A(1:2, :); % Specific row, : stands for all column
E = max(A, [], 2);
% The third arg stands for dimension=2, which means row
% Note: MATLAB is NOT row-major
F(1:2:5, 1:2:5) = A'; % Special indexing can be used in assignment
% Note: if not specified, the default value for an element is 0
disp(B);disp(C);disp(D);disp(E);disp(F);
```

Output:

```
>>  1    3

    7    9



    1    8

    7    3



    1    3    2

    8    4    6



    3

    8

    9



    1    0    8    0    7

    0    0    0    0    0

    3    0    4    0    9

    0    0    0    0    0

    2    0    6    0    5
```

(Next Page)

Problem 4 Solution

```
clear;clc;
% Some test cases:
fprintf("%d %d %d %d %d %d", my_gcd(0, 2), my_gcd(2, 0), my_gcd(0, 0), ...
    my_gcd(205, 25), my_lcm(12, 44), my_lcm(16, 404));
function out=my_gcd(a,b)
% Note: Here, my_gcd() behaves NOT the SAME as gcd()
if (a==0 || b==0)
    % Special judge for the case that a=0 or b=0
    % Return "Not A Number" instead of brutally throw an error
    out = nan;
    return
end
while b>0 % Continually calculates the reminder
    % If a>b the first loop will just swap them so it doesn't matter
    c = rem(a, b); % The Euclid's way
    a = b;
    b = c;
end
out = a;
end
function out=my_lcm(a,b)
if round(a) ~= a || round(b) ~= b || a < 1 || b < 1
    error('Input arguments should be postive integers.');
end
% Error handling, to eliminate illegal arguments(not an postive integer)
% Implicitly handled the case of insufficient parameters
out = a/my_gcd(a, b)*b; % Divided by common factor generated by my_gcd()
end
```

Output:

```
>>2 2 0 5 132 1616
```

Problem 5 Solution

```
clear;clc;
disp(lengthOnes('110100111'));
function y = lengthOnes(x)
y = 0;
current = 0;
for i=1:length(x)
    if x(i)=='1'
        current = current + 1; % Current consecutive times
    else
        y = max(y, current); % Update the answer
        current = 0; % Reset counter
    end
end
y = max(y, current); % Update the remained value in counter "current"
end
```

Example Output:

```
3
```

(End)