```
% MultiTable.m
% Patrick Utz, 2/23/18, 7.1

% Problem: Create a MATLAB script file to print multiplication
% table in the format shown in exercise 16 of Chapter 5. Your program
% should prompt the user to enter a positive integer number N, as the
% number of rows in the result. The example given in the book shows
% the result when N=5. Your program should be able to perform error-
% checking for the user inputted N value. If the user inputs a value
% that is not a positive integer, print an error message ?Invalid
% N value!? and prompt the user to enter N again. If the user could
% not enter a valid value of N for 5 times, print an error message
% ?You have entered invalid values for 5 times!?, and stop the program.
% Test your program using the following two sequences of input:
% 1) -5; 3.5; 0; 10. 2) -1, -2.5; 0; 3.5; -2. Attach your results.

% Variables: count = used to keep count of how many times user has
% inputted wrong data, N = the input of how many rows to make the
% matrix out of, array = the multiplication array

clear
count = 0;
N = input('Please enter a positive integer for the number of rows: ');
array = [];
for k = 1:4
   if N >= 1 && rem(N,1) == 0
      for l = 1:N
         fprintf('%d ', l:l:(l*l));
         fprintf('\n')
      end
      break;
   else
      fprintf('Invalid N value!\n');
      N = input('Please enter a positive integer: ');
      count = count + 1;
   end
end

if count == 4
   fprintf('You have entered invalid values for 5 times!\n');
else
   disp(array);
end
```

```
>> MultiTable
Please enter a positive integer for the number of rows: -5
Invalid N value!
Please enter a positive integer: 3.5
Invalid N value!
Please enter a positive integer: 0
Invalid N value!
Please enter a positive integer: 10
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25
6 12 18 24 30 36
7 14 21 28 35 42 49
8 16 24 32 40 48 56 64
9 18 27 36 45 54 63 72 81
10 20 30 40 50 60 70 80 90 100

>> MultiTable
Please enter a positive integer for the number of rows: -1
Invalid N value!
Please enter a positive integer: -2.5
Invalid N value!
Please enter a positive integer: 0
Invalid N value!
Please enter a positive integer: 3.5
Invalid N value!
Please enter a positive integer: -2
You have entered invalid values for 5 times!
```

```
% geomser.m
% Patrick Utz, 2/23/18, 7.2

% Problem: Create a MATLAB function called ?geomser? to calculate the
% sum of the following geometric sequence.1+r+r2 +r3 +r4 +...+rn
% Your function should receive two input arguments, r and n, where
% r is a real number, and n is a positive integer. Your function should
% return the sum as the output argument if the inputs are valid.
% Otherwise, it should return a 0. Use the following input values to
% test your function. Attach your results.(1)r=1,n=5;(2)r=0.5,n=9;(3)
% r=0.3,n=9.5;(4)r=2,n=-5

% Variables: total = output sum of function, r = real number r for
% series, n = positive integer for exponent of series, tempArray =
% temporary array used to find the series array

function total = geomser(r,n)
% geomser calculates the sum of the series with respect to the given r
% a and n values
% Format of call: geomser( real number r, positive integer n )
% Returns the sum of the series or 0 if an invalid value is inputted

if isreal(r) && (n >= 1 && rem(n,1) == 0)
   tempArray = r.^(0:n);
   total = sum(tempArray);
else
   total = 0;
end
```

```
>> geomser(1,5)
ans =
     6
>> geomser(.5,9)
ans =
    1.9980
>> geomser(.3,9.5)
ans =
     0
>> geomser(2,-5)
ans =
     0
>>
```

% billing.m
% Patrick Utz, 2/23/18, 7.3

% Problem: Write a MATLAB function named billing, which takes
% a vector of all residents? electricity usages (in units) as input
% argument, and generate a vector of all residents? payments as output
% argument. To test your function, write another script file that
% calls the function billing and plot the payments versus the usages
% for the range of 0< =usages<=1500 units with step size of 1 unit.
% Attach your figures.

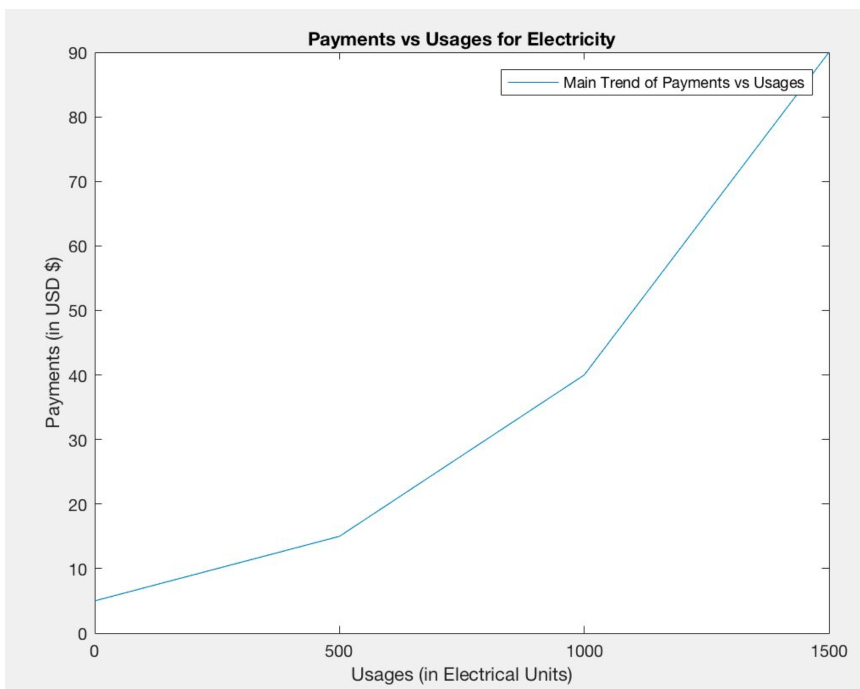
The electricity accounts of residents in a very small rural community are calculated as follows:

- If 500 or fewer units are used, the cost is 2 cents per unit.
- If more than 500, but not more than 1000, units are used, the cost is $10 for the first 500 units and 5 cents for every unit in excess of 500.
- If more than 1000 units are used, the cost is $35 for the first 1000 units plus 10 cents for every unit in excess of 1000.
- A basic service fee of $5 is charged no matter how much electricity is used.

% Variables: p = output payment vector, a = input usage vector

```
function p = billing(a)
% billing calculates the total amount of payment for electricity
% Format of call: billing( array of usage units )
% Returns an array of payments for the corresponding usage

for k = 1:length(a)
    if a(k) <= 500
        p(k) = ((a(k))*.02) + 5;
    elseif a(k) <= 1000
        p(k) = (((a(k))-500)*.05) + 15;
    else
        p(k) = (((a(k))-1000)*.10) + 40;
    end
end
```




```
clear
usages = 0:1500;
payments = billing(usages);
plot(usages,payments);
xlabel('Usages (in Electrical Units)');
ylabel('Payments (in USD $)');
title('Payments vs Usages for Electricity');
legend('Main Trend of Payments vs Usages');
```

% MAfilter.m
% Patrick Utz, 2/23/18, 7.4

% Problem: A moving-average filter is a basic digital signal
% processing tool to remove noise. Assume x is a vector of measured
% data samples with random noises, we can estimate the real data
% samples as a vector y as follows,

$$y(n) = \frac{1}{M}\sum_{l=0}^{M-1} x(n-l), \qquad for \quad n = 1 : length(x)$$

% That is, every sample in y is an M-point average over the last M-1
% samples and the current sample in x. This is called an M-th order
% moving-average filter. For the first M-1 samples, we only average
% from the first element in x up to the current value. Create a Matlab
% function named MAfilter to implement the M-th order moving average filter,
% the input argument should be a vector x and a filter order M, and the
% output should be the filtered vector y, with exactly same number of
% samples as in x. Use the following input to test your function. Attach
% your results.
% (1) x=[1,2,3,4,5]; M = 2;
% (2) x=[1,2.1,2.9,4.3,4.8];M=3;
% Note: You can NOT use any built-in filter functions in Matlab.

```
% Variables: y = output of filter in array form, x = input array,
% M = inputted filter order, n = reference for for loop

function y = MAfilter(x, M)
% MAfilter calculates the filtered array of x
% Format of call: MAfilter( input array x, filter order M )
% Returns the filtered array of x

for n = 1:length(x)
   y(n) = 0;
   for l = 0:(M-1)
      if (n-l) > 0
      y(n) = y(n) + x(n-l);
      else
         break;
      end
   end
   y(n) = (y(n)) * (1/M);
End
```

```
>> x = [1,2,3,4,5]
x =
     1     2     3     4     5
>> MAfilter(x,2)
ans =
    0.5000    1.5000    2.5000    3.5000    4.5000

>>  x= [1,2.1,2.9,4.3,4.8]
x =
    1.0000    2.1000    2.9000    4.3000    4.8000
>> MAfilter(x,3)
ans =
    0.3333    1.0333    2.0000    3.1000    4.0000
```

```
% denoising.m
% Patrick Utz, 2/23/18, 7.5

% Problem: Create a script file named denoising.m to do the following:
%     Step 1: generate the noisy vector x as follows:
%     1.First, generate a one-period sinusoidal signal s(n) with 100
%        linearly spaced samples from 0 to 2pi,
%     2.Then, generate 100 random noise samples w(n) ranging from -0.1
%        to 0.1.
%     3.Last, add each noise sample to the corresponding signal sample
%        to generate x.
%     Step 2: Call your MAfilter function written in Problem 7.4 with
%     two different M values: 2 and 5. Step 3: Plot the original clear
%     sinusoidal signal, the noisy signal, and both output signals
%     generated in Step 2 in one single figure. Use figure titles,
%     labels and legends to show the comparisons.
% Note: If you could not make MAfilter work in problem 7.4, you can
% use the built-in function filter() in the Digital Signal Processing
% toolbox to do Step 2 in the following way: y = filter([0.5, 0.5], 1, x)
% for M = 2 y = filter(0.2*ones(1,5), 1, x) for M=3

% Variables: c = x axis sample spacing, s = clean sin function, n =
% random noise vector, x = sin signal with noise, f1 = sin with noise
% and filter of M = 2 applied, f2 = sin with noise and filter of M =
% 5 applied

clear
c = linspace(0,2*pi,100);
s = sin(c);
n = rand(1,100)*(.1+.1)+(-.1);
x = s+n;
f1 = MAfilter(x,2);
f2 = MAfilter(x,5);

plot(c,s)
hold on
plot(c,x)
plot(c,f1)
plot(c,f2)
xlabel('Linear Sample Spacing')
ylabel('Signals as Functions of Samples')
```

```
title('Cleaning Signals Using Filters')
legend('Original Sinusoidal Signal','Signal with Noise', 'Filter Applied with M=2', 'Filter Applied with M=5')
```