E7: Introduction to Computer Programming for Scientists and Engineers
University of California at Berkeley, Spring 2017
Instructor: Lucas A. J. Bastien

Lab Assignment 03: Branching
– Solutions –
Version: release

Most of the time, if not always, there are more than one correct way to implement the functions for E7 lab assignments. This set of solutions proposes only one, or a few, of the possible implementations for each function.

For this assignment, the required submissions were:

- `my_consumer_helper.m`
- `my_water_quality.m`
- `my_array_resize.m`
- `my_sequence_id.m`
- `my_elevator.m`

1. Coupons (15 points)

The proposed solution for this question is:

my_consumer_helper.m

```
function [coupon, price] = my_consumer_helper(value)

% E7 Spring 2017, University of California at Berkeley.
% Solution function for question 1 of Lab 03.
%
% Version: release.

% Calculate the price that would result from applying each of the three
% coupons, separately for each coupon
price = 0.9 * value;
price2 = value - 25;
price3 = value - 50;

% Select the coupon 1 by default, and replace by the best coupon if
% necessary
coupon = 1;
if value >= 200 & value < 400 & price2 < price
    price = price2;
    coupon = 2;
```

```
elseif value >= 400 & price3 < price
    price = price3;
    coupon = 3;
end

end
```

The published test cases were:

```
>> % Published test case number 1 (1.5 point)
>> [coupon, price] = my_consumer_helper(60)

coupon =
     1
price =
    54

>> % Published test case number 2 (1.5 point)
>> [coupon, price] = my_consumer_helper(200)

coupon =
     2
price =
   175

>> % Published test case number 3 (1.5 point)
>> [coupon, price] = my_consumer_helper(390)

coupon =
     1
price =
   351

>> % Published test case number 4 (1.5 point)
>> [coupon, price] = my_consumer_helper(425)

coupon =
     3
price =
   375

>> % Published test case number 5 (1.5 point)
>> [coupon, price] = my_consumer_helper(550)

coupon =
     1
price =
   495
```

The hidden test cases are:

```
>> % Hidden test case number 1 (1.5 point)
>> [coupon, price] = my_consumer_helper(10)

coupon =
     1
price =
     9

>> % Hidden test case number 2 (1.5 point)
>> [coupon, price] = my_consumer_helper(215.50)

coupon =
     2
price =
  190.5000

>> % Hidden test case number 3 (1.5 point)
>> [coupon, price] = my_consumer_helper(400)

coupon =
     3
price =
   350

>> % Hidden test case number 4 (1.5 point)
>> [coupon, price] = my_consumer_helper(475.25)

coupon =
     3
price =
  425.2500

>> % Hidden test case number 5 (1.5 point)
>> [coupon, price] = my_consumer_helper(1025)

coupon =
     1
price =
  922.5000
```

2. Assessing water quality (15 points)

The proposed solution for this question is:

my_water_quality.m

```
1  function [score, warning] = my_water_quality(ph, do)
2
3  % E7 Spring 2017, University of California at Berkeley.
4  % Solution function for question 2 of Lab 03.
5  %
6  % Version: release.
```

```
7
8  % Set the PH score based on the value of the PH
9  if 6.5 <= ph & ph <= 7.5
10     score_ph = 5;
11 elseif 7.5 < ph & ph <= 8
12     score_ph = 4;
13 elseif 8 < ph & ph <= 8.5
14     score_ph = 3;
15 elseif 6 <= ph & ph < 6.5 | 8.5 < ph & ph <= 9
16     score_ph = 2;
17 elseif 5.5 <= ph & ph < 6 | 9 < ph & ph <= 9.5
18     score_ph = 1;
19 elseif 0 <= ph & ph < 5.5 | 9.5 < ph & ph <= 14
20     score_ph = 0;
21 else
22     score_ph = NaN;
23 end
24
25 % Set the DO score based on the value of the DO concentration
26 if 7 <= do & do <= 11
27     score_do = 5;
28 elseif 4 <= do & do < 7
29     score_do = 3;
30 elseif 2 <= do & do < 4
31     score_do = 1;
32 elseif 0 <= do & do < 2
33     score_do = 0;
34 else
35     score_do = NaN;
36 end
37
38 % Calculate the average score. Note that the following calculation will
39 % evaluate to NaN if and only if at least one of the two individual scores
40 % (PH score and DO score) is NaN (assuming that neither of these scores is
41 % infinite, because Inf+(-Inf) also evaluates to NaN)
42 score = (score_ph+score_do) / 2;
43
44 % Decide whether a warning should be issued
45 warning = isnan(score) | ph < 6.5 | ph > 8.5 | do < 4;
46
47 end
```

The published test cases were:

```
>> % Published test case number 1 (1.5 point)
>> [score, warning] = my_water_quality(7, 4)

score =
     4
warning =
  logical
```

```
   0

>> % Published test case number 2 (1.5 point)
>> [score, warning] = my_water_quality(6.2, 3)

score =
    1.5000
warning =
  logical
   1

>> % Published test case number 3 (1.5 point)
>> [score, warning] = my_water_quality(8.5, 2)

score =
     2
warning =
  logical
   1

>> % Published test case number 4 (1.5 point)
>> [score, warning] = my_water_quality(15, 10)

score =
   NaN
warning =
  logical
   1
```

The hidden test cases are:

```
>> % Hidden test case number 1 (1.5 point)
>> [score, warning] = my_water_quality(2.1, 2.5)

score =
    0.5000
warning =
  logical
   1

>> % Hidden test case number 2 (1.5 point)
>> [score, warning] = my_water_quality(8, 10.9)

score =
    4.5000
warning =
  logical
   0

>> % Hidden test case number 3 (1.5 point)
>> [score, warning] = my_water_quality(7, 15)
```

```
score =
   NaN
warning =
  logical
   1

>> % Hidden test case number 4 (1.5 point)
>> [score, warning] = my_water_quality(1, 11)

score =
    2.5000
warning =
  logical
   1

>> % Hidden test case number 5 (1.5 point)
>> [score, warning] = my_water_quality(6.75, 8)

score =
     5
warning =
  logical
   0

>> % Hidden test case number 6 (1.5 point)
>> [score, warning] = my_water_quality(0.0001, 0.0001)

score =
     0
warning =
  logical
   1
```

3. Array resizing (20 points)

The proposed solution for this question is:

my_array_resize.m

```
1  function array_out = my_array_resize(array_in, dimension)
2
3  % E7 Spring 2017, University of California at Berkeley.
4  % Solution function for question 3 of Lab 03.
5  %
6  % Version: release.
7
8  s = size(array_in);
9
10 if strcmpi(dimension, 'row')
11
12     array_out = [array_in; zeros(s)];
13
```

```
elseif strcmpi(dimension, 'col')

    array_out = [array_in, zeros(s)];

elseif strcmpi(dimension, 'both')

    array_out = [[array_in; zeros(s)], zeros(2*s(1), s(2))];

else

    array_out = array_in;

end

end
```

The published test cases were:

```
>> % Published test case number 1 (2.5 point)
>> my_array_resize([2, 3, 5; 4, 6, 7], 'row')

ans =
     2     3     5
     4     6     7
     0     0     0
     0     0     0

>> % Published test case number 2 (2.5 point)
>> my_array_resize([2, 3, 5; 4, 6, 7], 'col')

ans =
     2     3     5     0     0     0
     4     6     7     0     0     0

>> % Published test case number 3 (2.5 point)
>> my_array_resize([2, 3, 5; 4, 6, 7], 'both')

ans =
     2     3     5     0     0     0
     4     6     7     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0

>> % Published test case number 4 (2.5 point)
>> my_array_resize([2, 3, 5; 4, 6, 7], 'row and col')

ans =
     2     3     5
     4     6     7
```

The hidden test cases are:

```
>> % Hidden test case number 1 (2.5 point)
>> my_array_resize([-1, 0; 10, 2; 3, 4; 7, 3], 'row')

ans =
    -1     0
    10     2
     3     4
     7     3
     0     0
     0     0
     0     0
     0     0

>> % Hidden test case number 2 (2.5 point)
>> my_array_resize([-1, 0 10; 2, 3, 4; 7, 3, 1], 'col')

ans =
    -1     0    10     0     0     0
     2     3     4     0     0     0
     7     3     1     0     0     0

>> % Hidden test case number 3 (2.5 point)
>> my_array_resize([-1, 0; 10, 2; 3, 4; 7, 3], 'both')

ans =
    -1     0     0     0
    10     2     0     0
     3     4     0     0
     7     3     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0

>> % Hidden test case number 4 (2.5 point)
>> my_array_resize([-1, 0; 10, 2; 3, 4; 7, 3], 'just the columns')

ans =
    -1     0
    10     2
     3     4
     7     3
```

4. Sequences (25 points)

The proposed solution for this question is:

my_sequence_id.m

```
1 function [next_term] = my_sequence(a, b, c)
2
3 % E7 Spring 2017, University of California at Berkeley.
```

```
% Solution function for question 4 of Lab 03.
%
% Version: release.

% For each of the three sequences, if a, b, and c (in this order) can be
% three consecutive terms of this sequence, we will add the corresponding
% next term to the following array
next_terms = [];

% -> first sequence (Un)
if a + b == c
    next_terms(end+1) = b + c;
end

% -> second sequence (Vn)
if b - a == c
    next_terms(end+1) = c - b;
end

% -> third sequence (Wn)
if a * b == c
    next_terms(end+1) = b * c;
end

% Let us look at what we have in the array next_terms, and act accordingly:
% -> case 1: we did not find any possible next term
% -> case 2: we found at least one possible next term, and all the
%            possible next terms are the same
% -> case 3: we found at least two possible distinct next terms
if numel(next_terms) == 0
    next_term = NaN;
elseif all(next_terms == next_terms(1))
    next_term = next_terms(1);
else
    next_term = NaN;
end

end
```

The published test cases were:

```
>> % Published test case number 1 (2.5 points)
>> next_term = my_sequence_id(1, 1, 2)

next_term =
     3

>> % Published test case number 2 (2.5 points)
>> next_term = my_sequence_id(3, 7, 21)

next_term =
```

```
   147

>> % Published test case number 3 (2.5 points)
>> next_term = my_sequence_id(2, 2, 4)

next_term =
   NaN

>> % Published test case number 4 (2.5 points)
>> next_term = my_sequence_id(3, 5, 2)

next_term =
    -3
```

The hidden test cases are:

```
>> % Hidden test case number 1 (2.5 points)
>> next_term = my_sequence_id(100, 150, 250)

next_term =
   400

>> % Hidden test case number 2 (2.5 points)
>> next_term = my_sequence_id(10, -20, -30)

next_term =
   -10

>> % Hidden test case number 3 (2.5 points)
>> next_term = my_sequence_id(10, 20, 200)

next_term =
        4000

>> % Hidden test case number 4 (2.5 points)
>> next_term = my_sequence_id(-1, 1, -1)

next_term =
    -1

>> % Hidden test case number 5 (2.5 points)
>> next_term = my_sequence_id(1, 5, 0)

next_term =
   NaN

>> % Hidden test case number 6 (2.5 points)
>> next_term = my_sequence_id(0, 0, 0)

next_term =
     0
```

5. Elevators (25 points)

The proposed solution for this question is:

my_elevator.m

```
function [elevator] = my_elevator(location_caller, ...
                                  location_one, destination_one, ...
                                  location_two, destination_two)

% E7 Spring 2017, University of California at Berkeley.
% Function for question 5 of Lab 03.
%
% Version: release.

% Define useful logical quantities:
% -> same_floor_???: whether elevator ??? is at the same floor as the caller
% -> ???_moving: whether elevator ??? is moving
% -> ???_toward: whether elevator ??? is moving toward the caller
% -> d_???: distance between the location of elevator ??? and the caller
% -> dd_???: distance between the destination of elevator ??? and the caller
same_floor_one = (location_caller == location_one);
same_floor_two = (location_caller == location_two);
one_moving = ~isnan(destination_one);
two_moving = ~isnan(destination_two);
one_toward = one_moving & ...
    (destination_one-location_one)*(location_caller-location_one) > 0;
two_toward = two_moving & ...
    (destination_two-location_two)*(location_caller-location_two) > 0;
d_one = abs(location_caller - location_one);
d_two = abs(location_caller - location_two);
dd_one = abs(location_caller - destination_one);
dd_two = abs(location_caller - destination_two);

% Check the first condition
if same_floor_one & ~same_floor_two
    elevator = 1;
    return
elseif same_floor_two & ~same_floor_one
    elevator = 2;
    return
end

% Check the second condition
if one_toward & ~two_toward
    elevator = 1;
    return
elseif two_toward & ~one_toward
    elevator = 2;
    return
end

% Check the third condition
```

```
if ~one_moving & two_moving
    elevator = 1;
    return
elseif ~two_moving & one_moving
    elevator = 2;
    return
end

% Check the fourth condition
if one_moving & two_moving
    if dd_one < dd_two
        elevator = 1;
        return
    elseif dd_two < dd_one
        elevator = 2;
        return
    end
end

% Check the fifth condition
if d_one < d_two
    elevator = 1;
    return
elseif d_two < d_one
    elevator = 2;
    return
end

% If we reach this point of the code, then we should send elevator 1 (both
% elevators satisfied all the conditions, or both elevators satisfied none
% of the conditions)
elevator = 1;

end
```

The published test cases were:

```
>> % Published test case number 1 (2.5 points)
>> elevator = my_elevator(2, 1, -1, 2, 5)

elevator =
     2

>> % Published test case number 2 (2.5 points)
>> elevator = my_elevator(2, -2, 4, 1, 0)

elevator =
     1

>> % Published test case number 3 (2.5 points)
>> elevator = my_elevator(5, -2, NaN, 1, NaN)
```

```
elevator =
     2

>> % Published test case number 4 (2.5 points)
>> elevator = my_elevator(0, -1, -5, 2, 10)

elevator =
     1

>> % Published test case number 5 (2.5 points)
>> elevator = my_elevator(0, 3, 5, 2, NaN)

elevator =
     2

>> % Published test case number 6 (2.5 points)
>> elevator = my_elevator(10, 12, 10, 8, 10)

elevator =
     1
```

The hidden test cases are:

```
>> % Hidden test case number 1 (2.5 points)
>> elevator = my_elevator(-2, -2, 0, -3, NaN)

elevator =
     1

>> % Hidden test case number 2 (2.5 points)
>> elevator = my_elevator(50, 49, 48, 0, 10)

elevator =
     2

>> % Hidden test case number 3 (2.5 points)
>> elevator = my_elevator(4, 6, 10, 3, -2)

elevator =
     2

>> % Hidden test case number 4 (2.5 points)
>> elevator = my_elevator(0, 0, 1, 0, 1)

elevator =
     1
```