

Note: You can check these results by entering the expressions into MATLAB.

1.

```
>> x = 0:1:1
x =
    0    1
```
2.

```
>> x = [0 2 4 6 8 10];
>> y = x/max(x)
y =
    0    0.2000    0.4000    0.6000    0.8000    1.0000
```
3.

```
>> x = linspace(10,50,5);
>> y = ( min(x) < pi )
y =
    0
```

The expression `min(x)<pi` is *false* because `min(x)` is equal to ten. In MATLAB, as in most programming languages, *false* is the same as zero, and *true* is any value that is not zero.

4. The MATLAB function `fun` is defined as follows

```
function y = fun(a,b)
x = a^2 + b;
y = b - a;
```

What is result of executing the `disp` command when the following statements are entered in the command window?

- (a)

```
>> z = fun(1,2);
>> disp(x)
???
```

 Undefined function or variable 'x'.

The `x` variable in the `fun` function is local to that function: it only exists when the `fun` function is executing.

- (b)

```
>> z = fun(1,2);
>> x = 2*z;
>> disp(x)
2
```

5. The `myMax` function listed below correctly returns the value of the maximum element in a vector.

```
function xmax = myMax(x)

xmax = x(1);
for i=2:length(x)
    if x(i)>xmax
        xmax = x(i);
    end
end
```

For example,

```
>> x = -8:3;
>> myMax(x)
ans =
     3
```

Using `myMax` as a starting point, write a `myMinMax` function that computes and returns *both* `xmin` and `xmax`, where the value of the smallest (minimum) element is returned in `xmin`. For example, your `myMinMax` function should behave like this:

```
>> x = -8:3;
>> [a,b] = myMinMax(x)
a =
    -8
b =
     3
```

Do not use the built-in `min` or `max` functions.

Solution:

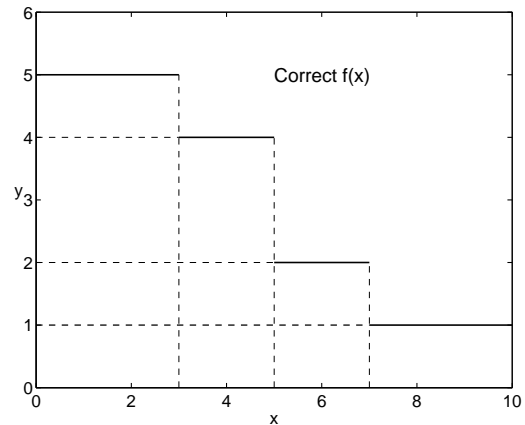
```
function [xmin,xmax] = myMinMax(x)

xmin = x(1);
xmax = x(1);
for i=2:length(x)
    if x(i)>xmax
        xmax = x(i);
    elseif x(i)<xmin
        xmin = x(i);
    end
end
```

6. The `stepFun` function contains errors. It is supposed to evaluate the discontinuous function depicted in the plot to the right.

```
function y = stepFun(x)
% stepFun Evaluate a discontinuous f(x)

if x<3
    y = 5;
elseif x>3
    y = 4;
elseif x>5
    y = 2;
elseif x>7
    y = 1;
end
```



The following MATLAB session shows the result of running `stepFun` for four different inputs.

```
>> stepFun(2)
ans =
     5
>> stepFun(4)
ans =
     4
>> stepFun(6)
ans =
     4
>> stepFun(8)
ans =
     4
```

Identify, and fix the error(s) in the `stepFun` function. You can indicate your changes directly in the listing of `stepFun`, or you can create a separate (handwritten) version of the function.

Although there are many ways to correct the function, solutions that are excessively complex or convoluted may not get full points even if they are correct. In other words, keep it simple.

Solution: The problem is caused by the inconsistent “direction” of the tests in the `if ...elseif` construct. The first test

```
if x<3
```

selects all values less than three (of course), but the next test

```
elseif x>3
```

selects all other cases except $x = 3$. In other words, the second test catches values of x that are greater than 5 and greater than 7 before the third and fourth tests can be evaluated. In fact, the

```
elseif x>5
```

and

```
elseif x>7
```

tests are *never* evaluated.

Here is one corrected implementation of the `stepFun`

```
function y = stepFun(x)
% stepFun Evaluate a discontinuous f(x)

if x<3
    y = 5;
elseif x<5
    y = 4;
elseif x<7
    y = 2;
else
    y = 1;
end
```