CS1112-206/211/212–Fall 2009
Section 5 Solution
Friday, October 2

The exercises for this week are relatively easy. This solution contains only the ideas and other points that might be misunderstood. Please refer to the solution on the course webpage for general MATLAB solutions.

1. First, the function should be called `Mid3` instead of `Min3` to reflect its objective. A simple solution to check that a variable x is the middle value is to make sure that x is at least another variable and at most the third variable. A solution to this approach is on the course webpage. One issue to note: We use "at least" and "at most," and not "more than" and "less than." Consider the solution where every `<=` is replaced with `<`. Then there is an instance where `Mid3` fails, namely, `Mid3(2,2,1)` returns 1 instead of 2.

   Despite its simplicity, this approach costs 8 comparisons. For efficiency, one might try to reduce the number of comparisons required to determine the middle value. Here is another approach which uses only 5 comparisons:

```
function y=Mid3(a,b,c)
% Find the middle value of the three given values a, b, and c.

if a<b
    if b<c          % a<b<c
        y=b;
    elseif a<c      % b>=c and a<c, so a<c<=b
        y=c;
    else            % a<b and a>=c, so c<=a<b
        y=a;
    end
else                % a>=b
    if a<c          % a>=b and a<c, so b<=a<c
        y=a;
    elseif b<c      % a>=c and b<c, so b<c<=a
        y=c;
    else            % a>=b and b>=c, so a<=b<=c
        y=b;
    end
end
```

   The idea here is to absorb the comparisons by utilizing the behavior of `if-elseif-else` construct.

   One issue to note: If two of the three values are the same, then the middle value is this duplicated value. It is easy to verify that the function above captures this using case-by-case analysis (there are only 3 cases).

2. The only trap for this exercise is to realize that the value `a` passed in is in degrees, but `sin` and `cos` take values in radians. Therefore, the first step is to convert the angle from d degrees to r radians using the formula $r = \frac{\pi d}{180}$. Then we can use `sin` and `cos` to evaluate the final values.

3. The complicated parts of this function are only graphics-related function calls and specifying what to plot. To specify what to plot, use the function `linspace`, which takes two arguments: the left and right endpoint, and an optional argument: the number of points in between (including the endpoints), which defaults to 100 if this is not specified. A solution that always works (and will not fail when run in any environment, e.g., with other figures open, etc.) is as follows:

```
function ShowSine(L,R)
% Produces a plot of the sine function across the interval [L,R]

close all;              % close all the opened figures
figure;                 % create a new figure (optional if this is the first figure)
x=linspace(L,R,100);    % domains (for the x-axis)
y=sin(x);               % values for the y-axis
plot(x,y);              % plot the graph
shg;                    % bring this figure to the top of all other windows
                        %  on the screen
```

4. Note that for this problem, it does not matter if the three sides form a triangle or not, because in case the sides do not form a triangle, the Pythagorean equality will not hold anyway and we will return 0. The only catch for this problem is that we do not know which side is the hypotenuse, so we need to try all of them and check if one equality is true, resulting in the solution on the course webpage. Note that we can use `if-else` construct instead of assigning it to `x` directly.

5. This is an example of treating a function as a black box (though we have been doing this for long with the MATLAB built-in functions such as `abs`). The idea is to transform the input to our function to a right format so that we can feed it to the black box (`MySin0`) and get the value we want. Because `MySin0` does really well if $|x| \leq 2\pi$, the input fed to it must be in this range. This is not too difficult for us to do so: just take the remainder of x when divided by 2π using `rem` function. Note that if x is negative, then the result will also be negative (try this for a few numbers, or read its description via `help rem`). Knowing this, we arrive at the code on the course webpage.

   Further exploration: functions `rem` and `mod` almost do the same thing, but with a subtle difference. Play around with them so you know which one to use when you write you own program.