CS1112-206/211/212–Fall 2009

Lab 2 Solution

Friday, September 11

Please note that solution programs are available on the course webpage in MATLAB format. To avoid inconsistency, we will avoid duplicating those programs here but instead concentrate on crucial parts of the exercises.

1 Minimum of a Quadratic

There are two parts in the assignment: Switch the values of `L` and `R` if necessary and reorder the three branches of the conditional statement in the original program.

1.1 Switching `L` and `R`

As mentioned in the section, there are those out-of-mind users who always give invalid input, either unintentionally or intentionally to break the program. In this case, the only illegal-input situation is when the interval given is invalid. That is, $L > R$. There are many approaches to handle this: from rejecting the input and terminate the program (most user-unfriendly approach) to fixing the input as appropriate and reasonable (my-program-is-smart approach) to asking the user to reenter the input (most user-friendly approach). In this exercise we take the second approach mentioned above.

Hence, the problem comes down to, "How do we write a program to swap the values of two variables?" As discussed, we cannot use two variables[1]; otherwise, the first line must look like this: `L = R;`. But we can see right away that after this statement is executed, we lose the value of L, with no hope of recovery. Therefore, we need a third, temporary variable that stores the value of L before it is gone: `tmp = L;`. Then, we can assign R to L and then tmp (which is the original value of L) to R. In MATLAB, this will look like

```
tmp = L;
L = R;
R = tmp;
```

1.2 Reordering the `if`-Statement Branches

We can reorder the branches of an `if` statement if and only if all possible cases tested by the `if` statement are handled. Because every possible value of x_c, the critical point, is handled—left of L, inside $[L, R]$ (maybe exactly at L or R), and right of R—we can reorder the branches.

The original program checks whether x_c is left of L first, but we want to check whether $L \leq x_c \leq R$ first in the modified version. Because every case is handled by this `if` statement, we simply move the second branch to the first, and we are done.

[1] In some computer languages, there *is* a way to swap variable values using no extra variable, but the procedure is much more complicated to come up with.

Note that a reordering is still possible even though not all cases are handled, but we need to make sure that the logical expression after each `if` or `elseif` means exactly as before. We have seen in lecture that the two following code snippets are not equivalent because they do not display the same message when $x = y$:

```
if (x<y)
  disp('lo');
else
  disp('hi');
```

```
if (x>y)
  disp('hi');
else
  disp('lo');
```

Therefore, when reordering branches in an `if` statement, make sure that the logical expressions in the new order still result in the original outcomes in *every* possible combination of the variables tested in those expressions.

2 Triangle

To determine that a triangle is scalene, we need to ensure that no two angles of the triangle have the same measure. That is, no matter how we pick the two angles, they should not be equal. Because there are three combinations possible to pick two angles, we need three comparisons: `a~=b && a~=c && b~=c`.

Now, if a triangle is not scalene, then it must be either equilateral or isoceles. To determine that a triangle is equilateral, we need to check that every angle of the triangle has the same measure. In other words, no matter how we pick two angles, they should be equal. That is, `a==b && a==c && b==c`; however, if we already know that $a = b$ and $a = c$, then it must be the case that $b = c$. Therefore, we do not need the third condition, even though it is still correct: `a==b && a==c` is enough.

There was a question whether we can write something like `a==b==c` instead of using an *and* operator. A simple answer here is that computers are, surprisingly, stupid, and can do only one comparison at a time. Having a statement like above will confuse the computer whether it should compare the first two first or the last two first. Same story goes with assignments[2].

3 When Do 3 Random Sticks Make a Triangle?

First, this should have read, "When do 3 random sticks make a triangle with positive area?" Note that a triangle with zero area is, for example, one with sides 2, 3, and 5. If we would like a triangle with positive area, then the sum of any two sides must exceed the length of the remaining side. (Otherwise, the endpoints of these sides cannot meet unless we bend or break the longest stick.) Hence, this reasoning results in the solution program.

One could reason more and arrive at a conclusion that it is sufficient to sum the two shortest sides and compare the sum with the longest side. In this case, we need to determine which two sides are shorest, or, equivalently, which side is longest. One possible solution is as follows:

[2] At least this is true in MATLAB. In some other languages, writing `x=y=2;` is legitimate.

```
% After this if, the longest side is at c.
if (a>=b)
    if (a>=c)   % a longest
        tmp=a;
        a=c;
        c=tmp;
    % Otherwise, c is longest and we are done
    end
else
    if (b>=c)   % b longest
        tmp=b;
        b=c;
        c=tmp;
    % Otherwise, c is longest and we are done
    end
end

% Now compare the sum with longest side
if (a+b>=c)
    disp('Yes');
else
    disp('No');
end
```

In both cases, the maximum number of comparisons is 3, but in the latter solution we only add numbers once.

4 Which Quadrant?

The trick of this question is to know what *rem* function does: return the remainder after division. Now, there have been some misunderstanding about what a *remainder* is. Let's discuss about that here to avoid further confusion. The following theorem should be useful:

Theorem: Let a and b be two integers where $b \neq 0$. Then there exist unique integers q and r such that $a = bq + r$, where $0 \leq r < |b|$. b is called the *divisor*, q is called the *quotient*, and r is called the *remainder*.

To put this in a familiar context, think about when we do a long division (without expanding to decimal points). We keep dividing until the last number left at the bottom is less than the divisor. This last number is the remainder, while the number at the top is the quotient.

Hence, even though the given angle A is less than 360 degrees, we can still use $\text{rem}(A, 360)$, whose result is indeed A.

There was a question whether `A=rem(A,360);` is legitimate due to a concern that the value of A will be lost. Indeed, the *original* value of A is lost, but we do not need that value if we have the remainder because we can use the remainder to determine the quadrant. Therefore, we can reuse

A by replacing its value with the remainder. In fact, this technique must be used for Project 1, Problem 1 to achieve the 3-variable solution.

5 Challenge Question

We already learned this in lecture last week, so this is not quite a challenge. There are a few solutions having two or three nested-`if` levels. If not already, can you strive for a solution that uses only two levels of nested `if`? (The solution on the course webpage is this way.)