Being concise and efficient: if, elseif, and boolean expressions

This is a reminder about certain nice properties of if-statements and how to cut down on superfluous code. Suppose you have a nonnegative ray angle $A$ in degrees. The following code determines in which quadrant $A$ lies:

```matlab
A = input('Input ray angle: ');
A = rem(A, 360); % Given nonnegative A, result will be in the interval [0,360)
if (A < 90) % first condition
    quadrant= 1;
elseif (A < 180) % second condition
    quadrant= 2;
elseif (A < 270)
    quadrant= 3;
else
    quadrant= 4;
end
fprintf('Ray angle %f lies in quadrant %d
', A, quadrant);
```

Notice that in the second condition, it is not necessary to check for $A\geq 90$ in addition to $A<180$ because the second condition, in the elseif branch, is executed only if the first condition evaluates to false. That means that by the time the computer gets to the second condition, it already knows that $A$ is $\geq 90$ so writing the compound condition $A\geq 90 \land A<180$ as the second condition would be redundant. Similarly, the concise way to write the third condition is to write only $A<270$ as above—unnecessary to write the compound condition $A\geq 180 \land A<270$. This is the nice (efficient) feature of “cascading” and “nesting.” If I do not cascade or nest, but instead use independent if-statements, then I must use compound conditions in some cases, as shown in the fragment below:

```matlab
A= rem(A, 360); %Given nonnegative A, result will be in the interval [0,360)
if (A < 90)
    quadrant= 1;
end
if (A >=90 \land A < 180)
    quadrant= 2;
end
if (A >=180 \land A < 270)
    quadrant= 3;
end
if (A >=270)
    quadrant= 4;
end
```

Efficiency is another consideration. The cascading style of code at the top of the page is more efficient than the fragment with the independent if-statements. To convince yourself, count the number of boolean operations (relational and logical) performed by the two fragments when $A$ is 300, 230, or 30.