Ok I figured I would get the ball rolling since no-one is posting.

#7 ok so i got zero for this because a NOR gate would add a and a not which would be 1 and then taking the not of that which would be 0, is this the correct way of doing this?

#9 I believe that this would be the rightmost structure because it is the and structure and just the not of it but I do not really understand how these work when there is not on the other side

#11 i am unsure of what to do as it is similar to #9. Do i just look at it as A not + B not equaling (not the sum of A not plus B not) and what would that end up being then

#14 how is this done exactly as do you just look at one row in the table and look at each answer and see if it works and if it does just keep checking down the table to see if every row works? I guess i just don't understand the method behind this

finally this is just sort of general but on Karnaugh maps do i just consider 0 to be not and 1 to be the regular letter? For example in number 16 i keep getting B*Cnot + Anot*C+ Anot*B and that is not right at all i just keep getting it wrong

Subject: Re: Digital Logic questions 7, 9, 11, 14, 16
Posted by sburkett on Thu, 08 Sep 2016 00:00:13 GMT

Thanks for taking initiative. I see great things in your future!

#7. Yes, zero is the answer.
#9. Yes, the rightmost structure implements NAND. The leftmost structure implements OR (use DeMorgan's Theorem to get A+B). The middle structure implements NOR (Put 2 bars over notA*notB and carry out DeMorgan's Theorem to lead you to not(A+B). Ask about this one in class on Friday.
#11. You want to get inputs that aren't inverted so you can see what LOGIC is implemented, not what logic symbol is in front of you. Using DeMorgan's Theorem, when you invert the quantity (notA + notB), you will get A AND B so AND is the answer.
#14. On this one, you could do a K-map but just looking at the answers, you might see that none of them appear to be what might result from simplifying so in that case, there is only one set of answers that has 4 terms (which you would get with no groupings and 4 terms) so I suggest starting with that one. This is (c) and is the answer; that expression describes every time the inputs give an output of 1. If that one didn't work, then you would try a K-map and see if any of the other answers worked. So yes, brute force will get the job done but if you started with the longest expression, you have the answer.
#16. Your K-map has 2 ones in [row 1, columns 2 and 3]. That grouping is B*notC (because A is changing, B is not changing and B is 1 and they are in the C=0 row so that is where notC comes
from). The other 2 ones are in [row 2, columns 1 and 2]. That grouping is not A*C (because A is not changing and it is 0; B is changing so throw it out; the ones are in C=1 row so that is where the C comes from). The answer to #16 is (a).

All good questions! Keep them coming!