HW5.2. Translating a Truth Table to a Boolean Circuit

Feel free to check out the guide that we have prepared to help you in this problem.

Consider the following truth table:

ABC|Q1.1Q1.2Q1.3Q1.4

000 1	1	0	0
001 1	0	1	0
010 0	0	0	1
011 0	0	1	1
100 1	1	1	0
101 1	0	0	0
110 0	0	1	0
11110	0	0	1

In this table, a 1 or 0 indicates an output which must be set, given the inputs A, B, and C.

Write boolean expressions for Q1.1, Q1.2, Q1.3, Q1.4 in terms of A, B, C. Partial credit will be awarded for answers not in simplest terms; for full credit, you must submit the simplest possible answer (i.e. the answer which requires the fewest operations). Note: The answer checker will only accept answers which are sequences of "A", "B", "C", "1", "0", and characters among "()*+!". Use ! for NOT, * for AND, + for OR, and () for grouping and precedence. Spaces are allowed, but will be ignored.

An example answer looks like this: (!A*!B)+(!C+!D)

This question will be scored as follows: First, your solution is compared against the reference solution on all valid inputs. If there is a mismatch, the solution is scored as 0. Otherwise, your score is computed as (r+1)/(s+1), where r is the number of boolean operators in our reference solution, and s is the number of boolean operators in the student submission. It is possible to get more than 100% on this question if your solution uses less boolean operators than the staff solution.



Try a new variant

Correct answer

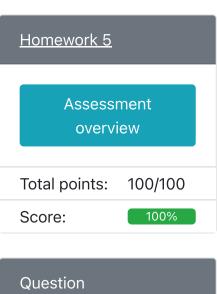
Feel free to check out the guide that we have prepared to help you in this problem.

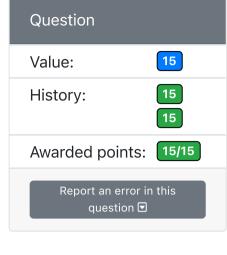
Q1.1: !B
Q1.2: !(B+C)
Q1.3: (!A*C)+(A*!C)

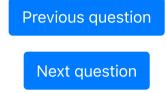
Q1.4: B*(!A+C)

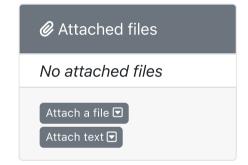
1: In this question, we notice that the output only depends on B. The answer is simply !B

2: Here, we notice that the output is only 1 if both B and C are 0. The expression would be !B*!C, which is simpler terms is !(B+C) using De Morgan's law.









- 3: For this one, we observe that the output becomes 1 when either A or C becomes 1, but not both. This is actually the expression for A xor C, that would have been the simplest implementation. However, since we cannot write xor for this question, we write the expanded form: (!A*C)+(A*!C)
- 4: There's no easily identifiable pattern for this question. Therefore, we resort to Boolean algebra by first writing out the terms where the output is 1. The full expression would be (!A*B*!C)+(!A*B*C)+(A*B*C). For the first two terms, !A*B are common, cancelling out the C term. The equation becomes: (!A*B)+(A*B*C). Then, we observe B is common, allowing us to rewrite the expression to be B*(!A+(A*C)). Using the uniting theorem, we can say that !A+ (A*C) = !A+C. Thus, we have the final simplified expression B*(!A+C)

