(I) (15pts) (Problem Formulation and Canonical Expression) A full adder inputs three bits \((a, b, c_{in})\) and outputs the sum bit and carry bit \((s, c_{out})\). Write the truth table of the full adder and the corresponding canonical product-of-maxterms expressions.

\[
\begin{array}{ccccc}
A & B & C_{in} & S & C_{out} \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[S = (a + b + c_{in})(a + b' + c_{in}')(a' + b + c_{in})(a' + b' + c_{in})\]

\[C_{out} = (a + b + c)(a' + b + c_{in})(a + b' + c_{in})(a + b + c_{in}')\]
(II) (10pts) (Laws and Theorems of Boolean Algebra) Specify the axioms or theorems for each step. Prove using Boolean algebra that
\[ a'b + a'c' + b'c' = a'b + b'c'. \]

\[
\begin{align*}
a'b + a'c' + b'c' & = a'b + a'c'(b+b') + b'c' \\
& = a'b + a'c'b + a'c'b' + b'c' \\
& = a'b + b'c' \quad (\text{Absorption of the first two and last two terms})
\end{align*}
\]

Can also use Consensus directly to prove the result.
(III) (10pts) (Laws and Theorems of Boolean Algebra) Specify the axioms or theorems for each step. Prove using Boolean algebra that

\[(a' + b')(a' + c')(b + c') = (a' + b')(b + c').\]

\[(a'+b')(a'+c')(b+c')\]

\[(a'+b')(b+c')\quad \text{(Applying Consensus)}\]

Note: There are several other ways of solving these two problems. All solutions were analysed while grading and people with explanations for their steps and who managed to prove the required result were given full credit.
(IV) (15pts) (Karnaugh Map) Use Karnaugh map to simplify function

\[ f(a, b, c) = \sum m(1, 2, 4) + \sum d(3, 6). \]

List all possible minimal two-level sum of products expressions. Show the Boolean expressions. No need for the logic diagram.

\[
F(a,b,c) = a'c + bc' + ac'
\]

\[
F(a,b,c) = a'c + a'b + ac'
\]
(V) (15pts) (Karnaugh Map) Use Karnaugh map to simplify function $f(a, b, c) = \sum m(3, 4) + \sum d(0, 2)$. List all possible minimal two-level product of sums expressions. Show the Boolean expressions. No need for the logic diagram.

```
F(a,b,c) = (a' + b')(b + c')
```
(VI) (20pts) Universal Set of Gates: Check if the set in the following list is universal and explain your decision. Assuming constants 0 and 1 are available as inputs.
   i.  \{AND, OR\}

There is no way one can get a “NOT” gate by making use of AND and OR. Hence it's not an universal set of gates

   ii.  \{NAND\} => output = (A.B)' (for inputs A and B)

   NOT:
   
   Connect A and B together. => output = (A.A)' = A'
   Or make B=1 => output = (A.1)' = A'

   AND:

   You already have NOT gate made out of NAND.
   NAND and NOT give AND i.e., NAND = (A.B)' ---NOT--- = (A.B)'' = A.B

   OR:

   Use NOT you made above to get A' and B'
   NAND of A' and B' = (A'.B')' = (A + B)

   Hence NAND is an universal gate!

   iii. \{ f(x,y) \}, where \( f(x,y) = x' + y' \)
   The easiest way is to show that \( x' + y' = (x.y)' \) => NAND gate
   You already proved that NAND is an universal gate

   NOT:

   for \( y = 1, y' = 0 \) => \( f(x,y) = x' + 0 = x' \) which is NOT

   AND:
(x' + y')' + (x' + y')' = (x' + y')' = x.y

OR:

Hence it's an universal gate!

iv. \{f(x,y,z)\}, where \(f(x, y, z) = x + y'z'\).

NOT:

x = 0 and z = 0 => f(x,y,z) = 0 + y'.0' = y'

AND:

Using the NOT gate described above, y' can be made from y and similarly z' from z
Now using $y'$ and $z'$ instead of $y$ and $z$ and putting $x=0$,

$$f = 0 + y.z = y.z$$

OR

Using the NOT gate described above, $y'$ can be made from $y$ and putting $z = 0$,

$$f = x + y.0' = x + y$$

Hence it's an universal gate!
(VII) (15pts) Shannon's Expansion: Use Shannon expansion's to prove the following equality.

\[ ab + bc' + cd + bd' + bc = b + cd. \]

Expansion for \( c \):

\[
\begin{align*}
f(a,b,c,d) &= c.f(a,b,1,d) + c'.f(a,b,0,d) \\
f(a,b,1,d) &= ab + d + bd' + b \\
&= b(a +1) + d + bd' \\
&= b (d' +1) +d \\
&= b + d \\
f(a,b,0,d) &= ab + b + bd' \\
&= b(a + 1 + d') \\
&= b \\
f(a,b,c,d) &= c.(b +d) + c'.(b) \\
&= bc +cd + c'b \\
&= b (c+c') + cd \\
&= b + cd
\end{align*}
\]

You can expand using any variable.