Implementation of Boolean Functions using NAND/NOR Gates

Q. 1. Present an implementation of the following Boolean function using two-input NAND gates. Assume that all variables are available in uncomplemented form as well as complemented form. (Do not change the form of the function.)

\[ Z = (A' + B)(C + D')E + (F + G') \]

**Step 1:** Build the circuit using two-input AND gates and two-input OR gates.

**Step 2:** Introduce a “bubble” at the output point of each AND gate (to convert it into a NAND gate) and compensate that appropriately. (“Look forward” along the same line.)
Step 3: Enforce uniformity, i.e., if there is a gate having a “bubble” at just one input point, then introduce a “bubble” at the other input point too, and compensate that appropriately. (“Look backward” along the same line.)

Step 4: Introduce “bubbles” at the input points of each OR gate and compensate that appropriately. If necessary, complement the input literals.

Step 5: Carefully re-organize the circuit. (Note that an inverter is realizable using a NAND gate.) Final circuit in this case is now immediate.
Q. 2. Present a circuit that implements the following Boolean function using two-input NAND gates:

\[ f(A, B, C, D, E) = (A' + B)(CD + E). \]

Assume that the independent variables are available in both complemented form and uncomplemented form.

An AND-OR implementation of the given function is immediate.

Introduce a “bubble” at the output point of each AND gate (to convert it into a NAND gate) and compensate that by introducing a “bubble” along the same line ahead at the input point of the next gate or by introducing an inverter.
Enforce uniformity, i.e., if there is a gate that has a “bubble” at one its input points, then introduce a “bubble” at the other input point too, and compensate that appropriately.

Introduce “bubbles” at the input points of any remaining OR gate, and compensate that appropriately.

Recall that an OR gate that has “bubbles” at all of its input points is equivalent to a NAND gate. Accordingly, the final circuit is as follows:
Q. 3. Present a circuit that implements the following Boolean function using two-input NAND gates:

\[ f(A, B, C, D, E) = A + (B' + C)(D' + BE'). \]

Assume that the independent variables are available in both complemented form and uncomplemented form.

Step 1: Obtain an AND-OR implementation of the given function:

```
A
B'
C
D'
B
E'
```

Step 2: Introduce a “bubble” at the output point of each AND gate (to convert it into a NAND gate) and compensate that by introducing a “bubble” along the same line ahead at the input point of the next gate.

```
A
B'
C
D'
B
E'
```

Step 3: Enforce uniformity, i.e., if a gate has a “bubble” at one of its input points, then each of the remaining input points of that gate also receives a bubble; compensation is achieved by either complementing the literal preceding that “bubble” or by introducing another “bubble” at the output point of the immediately preceding gate.
Step 4: Introduce a “bubble” at each input point of each OR gate that has not yet received a “bubble,” and compensate that appropriately.

Step 5: Recall that an OR gate that has “bubbles” at all of its input points is equivalent to a NAND gate. Accordingly, the final circuit is as follows:
Q. 4. Present a realization of the following function using two-input NAND gates:

\[ f(a, b, c, d) = abc + ad + c'd'. \]

Assume that each variable is available in its complemented form as well as uncomplemented form. Use as few gates as possible.

Note that

\[ f(a, b, c, d) = abc + ad + c'd' \]
\[ = a(bc + d) + c'd'. \]

Implementation follows.

Note: Details are left to the reader.
Q. 5. Present a realization of the following function using two-input NOR gates:

\[ g(a, b, c, d, e, f) = ae + bde + bcef. \]

Assume that each variable is available in its complemented form as well as uncomplemented form. Use as few gates as possible.

Note that

\[
  g(a, b, c, d, e, f) = ae + bde + bcef = (a + bd + bcf)e = (a + b(d + cf))e.
\]

Implementation follows.

Note: Details are left to the reader.