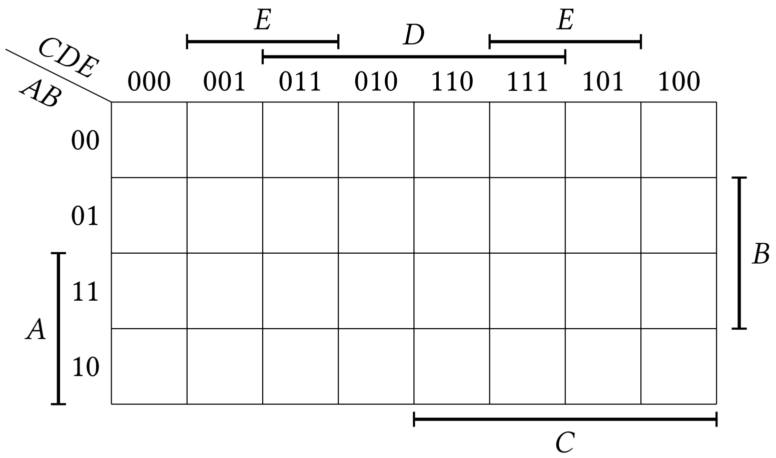
1. Perform the following steps:
   (a) Calculate the SHA-256 hash h of the string $s =$ `"Your Full Name HW3"` (substitute your full name as in the scores table, without quotes, with all spaces, encoded in UTF-8). Convert hash h to a 256-bit binary string b (prepend leading zeros if necessary). Cut the binary string b into eight 32-bit slices $r_1, \dots, r_8$, *e.g.*, $r_2 = b_{33..64}$. Xor all slices into a 32-bit string $d = r_1 \oplus \dots \oplus r_8$.
   (b) Draw the Karnaugh map (use a template below) for a function $f(A, B, C, D, E)$ defined by the truth table d (MSB corresponds to $\mathbb{0}$, LSB to $\mathbb{1}$). Use it to find the number of prime implicants[1].



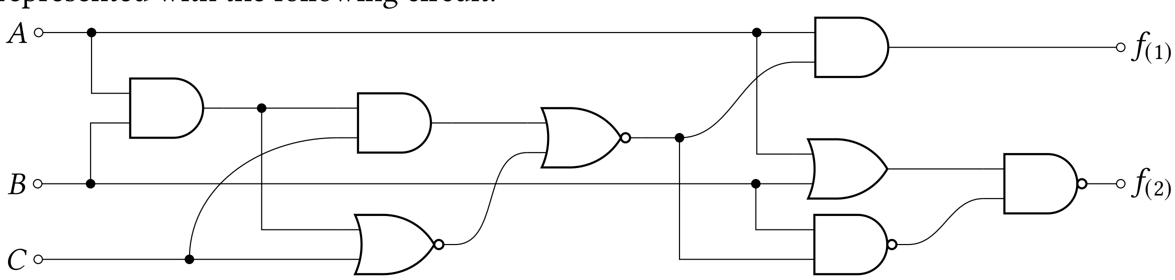
2. For each given function f_i of 4 arguments, draw the Karnaugh map and use it to find BCF, minimal DNF, and minimal CNF. Additionally, construct ANF (Zhegalkin polynomial) using either the tabular ("triangle") method or the Pascal method — use each method at least once.

> **Note:**WolframAlpha interprets the query "n-th Boolean function of k variables" in a reverse manner. In order to employ WolframAlpha properly, manually flip the truth table beforehand, *e.g.*, the correct query for $f_{10}^{(2)}$ is "5th Boolean function of 2 variables", which gives $f_{10}^{(2)} = \neg x_2$, since $\texttt{rev}(1010_2) = 0101_2 = 5_{10}$.

   (a) $f_1 = f_{47541}^{(4)}$
   (b) $f_2 = \sum m(1, 4, 5, 6, 8, 12, 13)$
   (c) $f_3 = f_{51011}^{(4)} \oplus f_{40389}^{(4)}$
   (d) $f_4 = A\overline{B}D + \overline{A}\,\overline{C}D + \overline{B}C\overline{D} + A\overline{C}D$

3. Convert the following formulae to CNF.
   (a) $X \leftrightarrow (A \land B)$
   (b) $Z \leftrightarrow \bigvee_i C_i$
   (c) $D_1 \oplus \dots \oplus D_n$
   (d) $\texttt{majority}(X_1, X_2, X_3)$ [2]
   (e) $R \rightarrow (S \rightarrow (T \rightarrow \bigwedge_i F_i))$
   (f) $M \rightarrow (H \leftrightarrow \bigvee_i D_i)$

4. Compute the truth table for the function $f \colon \mathbb{B}^3 \to \mathbb{B}^2$ (with the semantics $\langle A, B, C \rangle \mapsto \langle f_{(1)}, f_{(2)} \rangle$) represented with the following circuit.



[1] Here, consider only implicants represented as product terms.
[2] Majority function is a Boolean function that is 1 iff the majority (more than half) of the inputs are 1.

5. Show — without using Post's criterion — that the Zhegalkin basis $\{\oplus, \wedge, 1\}$ is functionally complete.

6. For each given system of functions F_i, determine whether it is functionally complete. For each basis F_i, use it to rewrite the function $g(A, B, C) = A \to ((\neg A \oplus B) \wedge \neg C)$. Draw a schema (Boolean circuit) for each resulting formula.
   (a) $F_1 = \{\wedge, \vee, \neg\}$
   (b) $F_2 = \{f_{14}^{(2)}\}$
   (c) $F_3 = \{\to, \not\to\}$
   (d) $F_4 = \{1, \leftrightarrow, \wedge\}$

7. Construct a minimal (in terms of the number of gates) Boolean circuit that implements the conversion of 4-bit binary numbers to Gray code, *i.e.* the function $f\colon \mathbb{B}^4 \to \mathbb{B}^4$ with the semantics $(b_3, b_2, b_1, b_0) \mapsto (g_3, g_2, g_1, g_0)$, *e.g.*, $\texttt{0000}_2 \mapsto \texttt{0000}_{\text{Gray}}$, and $\texttt{1001}_2 \mapsto \texttt{1101}_{\text{Gray}}$. Use only `NAND` and `NOR` logic gates.

8. Consider a Boolean function $f\colon \mathbb{B}^3 \to \mathbb{B}$ defined as follows: $f(x, y, z) = \left\{\begin{smallmatrix} x & \text{if } z=0 \\ y & \text{if } z=1 \end{smallmatrix}\right.$. Construct a formula for it using the standard Boolean basis $\{\wedge, \vee, \neg\}$.

9. Binary Decision Diagram is a representation of a Boolean function as a directed acyclic graph, which consists of *decision* nodes and two *terminal* nodes (0 and 1). Each decision node is labeled by a Boolean variable x_i and has two child nodes called *low* and *high*. The edge from node to a low (high) child represents an assignment of the value `FALSE` (`TRUE`, respectively) to variable x_i. A path from the root node to the 1-terminal (0-terminal) represents an assignment for which the represented Boolean function is true (false, respectively).
   BDD is called *ordered* if different variables appear in the same order on all paths from the root. For example, if the natural order $x_1 \prec x_2 \prec \cdots \prec x_n$ is used, the root is marked with the variable x_1, its children with x_2, *etc.* Note that some variables in the order can be skipped, if necessary.
   For each given function f_i, construct an Ordered Binary Decision Diagram using the natural order. Determine whether the OBDD can be reduced by using a different variable order — if so, draw it.
   (a) $f_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$
   (b) $f_2 = \texttt{majority}(x_1, \ldots, x_5)$
   (c) $f_3 = \sum m(1, 2, 5, 12, 15)$
   (d) $f_4 = x_1x_4 + x_2x_5 + x_3x_6$