In this problem, you will be asked questions pertaining to different representations of Boolean functions. These go by an alphabet soup of acronyms:

1. Truth Tables
2. Disjunctive Normal Form (DNF)
3. Conjunctive Normal Form (CNF)
4. Exclusive-OR Normal Form (XNF)
5. Binary Decision Diagrams (BDDs)
6. Boolean Satisfiability (SAT)

You are probably most familiar with Boolean functions expressed in terms of the operations AND, OR and NOT. For instance, consider the function specified by the following truth table.

x_1	x_2	x_3	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

The function can be represented as

$$f = \bar{x}_1(\bar{x}_2 + \bar{x}_3).$$

Here addition denotes OR, multiplication denotes AND, and an overbar denotes NOT.

DNF and CNF are canonical forms of sum-of-products and product-of-sums expressions, respectively. (You are expected to know what these forms are.) For the function above, these are

$$f = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3,$$

and

$$f = (x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + x_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3),$$

respectively.

A less common representation is based on the AND and exclusive-OR (XOR) operations. We will denote the XOR operation with $\oplus$. As early as 1929, Zhegalkin showed that this representation is canonical: if we multiply out all parentheses, perform the simplifications $x \oplus x = 0$ and $x \cdot x = x$, and sort the product terms, the resulting expression is unique. Accordingly, we'll call the representation XNF, for XOR Normal Form. (This isn't a standard term for the representation. It is sometimes known as the Reed-Muller form). In this representation, the function above is

$$f = 1 \oplus x_1 \oplus x_2x_3 \oplus x_1x_2x_3.$$

(Note that $1 \oplus x = \bar{x}$.)

The XNF representation has distinct advantages when manipulating expressions algebraically. Since it is canonical, we need not concern ourselves with simplifying the expressions, as we would working with sum-of-products or products-of-sums representations. Furthermore, unlike those representations, the dependence of a function on its variables is explicit in XNF. If a variable appears in an expression, then there exists some assignment of values to the other variables such the value of the expression depends on the value of that variable.

First proposed in 1959 by Lee, binary decision diagrams (BDDs) were popularized in 1986 through a seminal paper by Bryant. A BDD consists of a directed graph in which nodes either have associated input variables or else are designated as a constant nodes ("0" or "1"). To evaluate a function one begins at a designated source node and follows a path dictated by the values of the variables until one arrives at one of the two constant nodes. The value of this constant node specifies the value of the function. The BDD for the function above is shown in Figure 1.

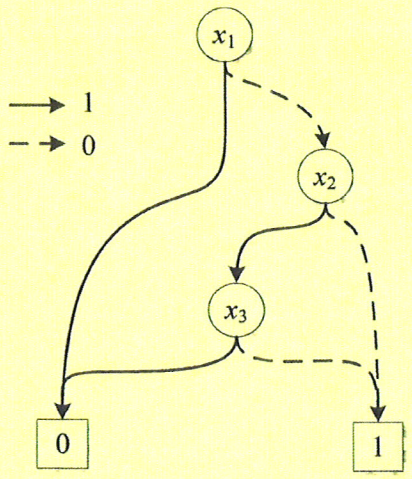

Figure 1: A binary decision diagram (BDD).

Although comparable in size to a truth table in the worst case, BDDs are surprisingly compact for many of the Boolean functions encountered in practice. This is due to the

fact that BDDs can often be reduced in size by *collapsing* redundant nodes and *merging* equivalent nodes.

Boolean satisfiability (SAT) is not, technically speaking, a representation for Boolean functions. Rather, SAT-based techniques, based on heuristic solutions to the Boolean satisfiability problem, are an algorithmic technique widely used in logic synthesis and verification. SAT-based analysis begins with a circuit structure and proceeds by packaging the Boolean function that it computes in CNF. This is passed to heuristic algorithms known as SAT solvers. If the solver returns "UNSAT," this means that there is no satisfying assignment to the formula. Otherwise, the solver returns "SAT" along with a satisfying assignment. For instance, consider the circuit in Figure 2. The corresponding CNF formula is:

$$(\bar{x}_2 + y)(\bar{x}_3 + y)(x_2 + x_3 + \bar{y})(x_1 + \bar{h})(y + \bar{h})(\bar{x}_1 + \bar{y} + h)(h).$$

The solver would return SAT. A satisfying assignment for this formula is $x_1 = x_2 = x_3 = y = h = 1$.

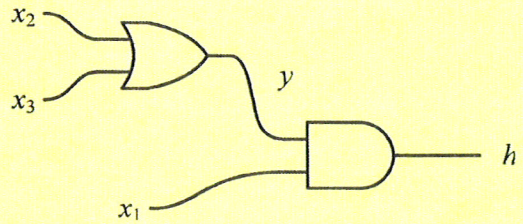

Figure 2: A circuit illustrating Boolean satisfiability.

In theory, SAT-based algorithms can take time that is exponential in the number of variables to complete. In practice, they have shown themselves to be remarkably efficient.

Questions

Consider the functions:

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1x_2x_3 + \bar{x}_3(\bar{x}_2 + \bar{x}_1) + \bar{x}_1\bar{x}_2, \\ g(x_1, x_2, x_3) &= \bar{x}_1\bar{x}_2x_3 + \bar{x}_3(x_1\bar{x}_2 + \bar{x}_1x_2), \\ h(x_1, x_2, x_3, x_4) &= x_1 + x_2 + x_3 + x_4. \end{aligned}$$

1. Write truth tables for these functions [10%].
2. Write these functions in Disjunctive Normal Form (DNF) [10%].
3. Write these functions in Conjunctive Normal Form (CNF) [10%].
4. Write these functions in Exclusive-OR Normal Form (XNF) [30%].
5. Draw fully reduced Binary Decision Diagrams (BDDs) for these functions. Use the variable ordering x_1, x_2, x_3, x_4 [20%].
6. Consider the following functions:

$$\begin{aligned} f_1(x_1, x_2, x_3) &= \bar{x}_1\bar{x}_3 + x_2x_3 + x_1\bar{x}_2 \\ f_2(x_1, x_2, x_3) &= \bar{x}_2\bar{x}_3 + \bar{x}_1x_2 + x_1x_3 \end{aligned}$$

   Draw circuits corresponding to the these functions. Formulate a new circuit corresponding to the question: are these two functions equivalent? Create a function for the new circuit, packaged for Boolean Satisfiability (SAT). What would the outcome of SAT be? Are the two functions equivalent? [20%].