A BDD (Binary Decision Diagram) is a binary expression that has one of the following 3 forms: \( \top \), \( \bot \), if variable then BDD else BDD fi. For example,

\[
\text{if } x \text{ then if } a \text{ then } \top \text{ else } \bot \text{ else if } y \text{ then if } b \text{ then } \top \text{ else } \bot \text{ else } \bot \text{ fi}
\]

is a BDD. An OBDD (Ordered BDD) is a BDD with an ordering on the variables, and in each if then else fi, the variable in the if-part must come before any of the variables in its then- and else-parts (“before” means according to the ordering). For example, using alphabetic ordering for the variables, the previous example is not an OBDD, but

\[
\text{if } a \text{ then if } c \text{ then } \top \text{ else } \bot \text{ else if } b \text{ then if } c \text{ then } \top \text{ else } \bot \text{ else } \bot \text{ fi}
\]

is an OBDD. An LBDD (Labeled BDD) is a set of definitions of the following 3 forms:

\[
\text{label} = \top
\]

\[
\text{label} = \bot
\]

\[
\text{label} = \text{if variable then label else label fi}
\]

The labels are separate from the variables; each label used in a then-part or else-part must be defined by one of the definitions; exactly one label must be defined but unused. The following is an LBDD.

\[
\text{true} = \top
\]

\[
\text{false} = \bot
\]

\[
\text{alice} = \text{if } b \text{ then true else false fi}
\]

\[
\text{bob} = \text{if } a \text{ then alice else false fi}
\]

An LOBDD is an LBDD that becomes an OBDD when the labels are expanded. The ordering prevents any recursive use of the labels. The previous example is an LOBDD. An RBDD (Reduced BDD) is a BDD such that, in each if then else fi, the then- and else-parts differ. An RLOBDD is reduced, labeled, and ordered. The previous example is an RLOBDD.

(a) Express \( \neg a \), \( a \land b \), \( a \lor b \), \( a \Rightarrow b \), \( a = b \), \( a \neq b \), and \( \text{if } a \text{ then } b \text{ else } c \text{ fi as BDDs.} \)

\[
\neg a \text{ is } \text{if } a \text{ then } \bot \text{ else } \top \text{ fi}
\]

\[
a \land b \text{ is } \text{if } a \text{ then if } b \text{ then } \top \text{ else } \bot \text{ else } \bot \text{ fi}
\]

\[
a \lor b \text{ is } \text{if } a \text{ then } \top \text{ else if } b \text{ then } \top \text{ else } \bot \text{ fi}
\]

\[
a \Rightarrow b \text{ is } \text{if } a \text{ then if } b \text{ then } \top \text{ else } \bot \text{ else } \top \text{ fi}
\]

\[
a = b \text{ is } \text{if } a \text{ then if } b \text{ then } \bot \text{ else } \bot \text{ else if } b \text{ then } \bot \text{ else } \top \text{ fi}
\]

\[
a \neq b \text{ is } \text{if } a \text{ then if } b \text{ then } \bot \text{ else } \bot \text{ else if } b \text{ then } \bot \text{ else } \top \text{ fi}
\]

\[
\text{if } a \text{ then } b \text{ else } c \text{ fi is } \text{if } a \text{ then if } b \text{ then } \bot \text{ else } \bot \text{ else if } c \text{ then } \top \text{ else } \bot \text{ fi}
\]

(b) How can you conjoin two OBDDs and get an OBDD?

\[
\text{If one of them is } \top \text{ then the answer is the other one. If one of them is } \bot \text{ then the answer is } \bot \text{. If both of them are ifs with the same variable as if-part, say } \text{if } a \text{ then OBDD0 else OBDD1 fi and if } a \text{ then OBDD2 else OBDD3 fi, then the answer is if } a \text{ then OBDD0} \land \text{OBDD2 else OBDD1} \land \text{OBDD3 fi where the same procedure is used to find the then- and else-parts. If both of them are ifs with different variables as if-parts, and the one with the earlier variable is if } a \text{ then OBDD0 else OBDD1 fi, and the other one is OBDD2, then the answer is if } a \text{ then OBDD0} \land \text{OBDD2 else OBDD1} \land \text{OBDD2 fi where the same procedure is used to find the then- and else-parts.}
\]

(c) How can you determine if two RLOBDDs are equal?

\[
\text{Two RLOBDDs are equal if and only if they are identical except for a one-for-one substitution of labels and reordering of equations.}
\]

(d) How can we represent an RLOBDD in order to determine efficiently if an assignment of values to variables satisfies it (solves it, gives it value \( \top \))?