We will implement the unification algorithm and try to implement a term rewriting system if time permits.

We will first define the data types we need.

1. Define a suitable type `vname` for variables
2. Define a suitable type `term` for terms
3. Define a suitable type `subst` for substitutions (possibly a `(vname * term)` list)

We need a few functions to aid us.

4. Define a function `contains` which checks if a variable `x` is present in the domain of a substitution. (type: `vname -> subst -> bool`)
5. Define a function `substitute` which return the correct substitute for a variable `x`. (type: `subst -> vname -> term`)
6. Define a function `lift_subst` which return the correct substitute for a term `t`. (type: `subst -> term -> term`)
7. Define a function `occurs` which checks if a variable occurs in a term `t`. (type: `vname -> term -> bool`)

We will now go on to implement the unifier. An instance to the problem will be a list of pairs of terms `{(s_1, t_1), ..., (s_n, t_n)}`. The output will be a substitution \( \phi \) such that \( \phi(s_i) = \phi(t_i) \). Given a unification problem \( C \) consisting of a head \((s, t)\) and tail \(C'\), there are some cases to consider

- **Delete rule:** If \( s \) and \( t \) are are equal, discard the pair, and unify \( C' \).
- **Eliminate rule:** If \( s \) is a variable, and \( s \) does not occur in \( t \), substitute \( s \) with \( t \) in \( C' \) to get \( C'' \). Let \( \phi \) be the substitution resulting from unifying \( C'' \). Output \( \phi \) updated with \( s \rightarrow \phi(t) \).
- **Orient rule:** If \( t \) is a variable and \( s \) is not, then discard \((s, t)\), add \((t, s)\) to \( C'\), and unify the result.
- **Decompose rule:** If \( s \) and \( t \) are non variable terms, assert that the roots are the same, discard this pair and insert the pairs coming from the successors. that is: \((f(t_1 \ldots t_n), f(u_1 \ldots u_n)) :: C' \rightarrow (t_1, u_1) :: \ldots :: (t_n, u_n) :: C'\).
• If $C$ is empty, then we return the identity substitution.

• If none of the above cases apply, it is a unification error (your unify function should return bottom ($\bot$) or raise an exception in this case).

8. Implement the algorithm to unify.

We will consider the matching problem now. The input is again a list of pairs of terms as for unification. However the $\phi$ we compute need to satisfy $\phi(s_i) = t_i$.


Now we will implement a term rewrite system. Let $R$ be a set of rewrite rules (given as (term * term) list).

10. Implement a function \texttt{rewrite} $R$ $t$ which will perform a single $\rightarrow_R$ step at the root of $t$. (You will need to the match the left hand side of rule with $t$ using the matching algorithm in question 9.) Write a function \texttt{norm} $R$ $t$ which will compute an $R$-normal form for $t$. 