INTRO. TO LOGIC & FUNCT. PROG.

Consider the representation of "pre-terms" using the following data type definition

type *term* = V of *variable* | Node of *symbol** (*term* **list**);;

Choose suitable type representations for types *variable* and *symbol.*

1. *Given a signature* consisting of symbols and their arities (>= 0) in any suitable form -- either as a list of (symbol, arity) pairs, or as a function from symbols to arities, write a function ***check_sig*** that checks whether the signature is a valid signature (no repeated symbols, arities are non-negative etc.)
2. Given a *valid* signature (checked using ***check_sig***), define a function ***wfterm*** that checks that a given preterm is well-formed according to the signature.
3. Define functions ***ht***, ***size*** and ***vars*** that given a well-formed term, return its height(leaves are at height 0) , its size (number of nodes) and the set of variables appearing in it respectively. Use *map*, *foldl* and other such functions as far as possible wherever you use lists.
4. Define a suitable representation for substitutions.
5. Come up with an efficient representation of *composition of substitutions.*
6. Define the function ***subst*** that given a term *t* and a substitution *s,* applies the (Unique Homomorphic Extension of) *s* to *t.* Ensure that ***subst*** is efficiently implemented.
7. Define the function *mgu* that given two terms *t1* and *t2*, returns their most general unifier, if it exists and otherwise raises an exception *NOT_UNIFIABLE.*
8. For each of your programs define