Recall the small language for which we wrote a type checker using the predicate hastype(Gamma, E, T) which takes a set of type assumptions Gamma, an expression E, and a type T.    The predicate is true if the expression has the given type in the context of the type assumptions.

The object language consists of expressions of the form

* variables, written using the constructor v(\_)
* abstractions, written using the constructor abs(\_,\_), where the first argument will be a variable, and the second an expression
* applications, written using the constructor app(\_,\_\_), where both the arguments are expressions.

The object language’s type system consists of

* (unspecified) type variables,  represented using the meta-language PROLOG’s variables.
* base types such as t\_int, t\_bool, etc.
* function types (T1 -> T2), represented in PROLOG using the constructor  arrow(T1, T2).

Sets of type assumptions on variables are written as lists of (object language) variable-type pairs.

Now let us extend the language and its type-checker hastype(Gamma, E, T) to include the following:

* arithmetic expressions of the form: plus(E1,E2), times(E1, E2), which will be of type t\_int (if the sub-expressions are).
* boolean expressions of the form:  neg(E),  conj(E1, E2), disj(E1, E2), which will be of type t\_bool (if the sub-expressions are).
* comparison expressions of the form: leq(E1,E2), equals(E1, E2), geq(E1, E2), of type t\_bool for subexpressions of type t\_int.
* the expression of the form: unit, of type t\_unit.
* pair expressions of the form: pair(E1, E2) of type product(T1, T2)
* projection expressions of the form: proj1(E) and proj2(E), which expect E to be of a product type..
* injection expressions of the form: inl(E),  inr(E), of type sum(T1, T2)  assuming E is of type T1, respectively T2.
* case expressions of the form: case(E0, inl(X), E1, inr(Y), E2), where E0 is of a sum(T1, T2) type, and E1, E2 are of the same return type T, when X is of type T1 and Y of of type T2.
* conditional expression ifte(E0, E1, E2), where E0 is of  type t\_bool and E1, E2 are of the same type..

Provide sufficient examples of programs to show your type checker and type inference runs correctly.