Problem 1 – Syntax-Directed Translation Schemes

Assume the following partial grammar:

```
vardcl ::= idlist : type 
|        
idlist  ::= idlist, ID | ID 
type    ::= integer | real | double 
```

Write a syntax-directed translation scheme that stores the types of the variables in a symbol table. You may use pseudo code in your embedded actions. Assume that each ID has a pre-defined synthesized attribute `name` that contains its lexeme. The routine `insert(id, type)` inserts an identifier of a particular type into the symbol table. Use YACC-like notation (e.g. `$$.name` or `$1.type`).

Problem 2 – Type Systems

Assume a type system with the following inference rules

\[
Rule_{A1} : \quad \frac{E \vdash e_1 : integer \quad E \vdash e_2 : integer}{E \vdash (e_1 + e_2) : integer}
\]

\[
Rule_{A2} : \quad \frac{E \vdash e : integer}{E \vdash &e : pointer(integer)}
\]

\[
Rule_{A3} : \quad \frac{E \vdash e : pointer(integer)}{E \vdash *e : integer}
\]

Assuming that variable `a` and constant `3` are of type `integer`, use the inference rules to determine the types of the following expressions. Note: if a proof does not exist, the type system reports a type error.
1. &a
2. (&a + 3)
3. *a
4. &3
5. (*&a + 3)
6. &;&a