Problem 1a

* a and b are directly compatible with each other since they are both of type S.
* c is compatible with both a and b because T is a typedef for S, meaning they share the same underlying structure.
* d and e have the same structure as S (and therefore T), making them structurally equivalent to a, b, and c.

So, under strucutral equivalence, all the variables a, b, c, d, and e, are considered to have compatible types since they all share the same structure of an int followed by a double, regardless of the typedef names S and T.

Problem 1b

* a and b are compatible with each other since they are both explicitly declared as type S.
* c is not directly compatible with a and b under strict name equivalence, despite T being a typedef for S, because T is considered a different name.
* d and e are not compatible with a, b, or c, nor with each other under strict name equivalence, as they are declared with an anonymous structure that, while structurally identical to S and T, is considered a distinct type because it does not have a name.

So, under strict name equivalence, only variables declared with the exact same type name or typedef alias are considerred to have compatible types. Therefore, in this scenario:

* a and b are compatible with each other.
* c, d, and e are not considered compatible with any other variable under strict name equivalence due to the rules requiring exact name matches for compatibility.

Problem 1c

* a and b are equivalent to each other since they are of the same type S.
* c is equivalent to a and b because T is a typedef of S, and under structural equivalence, the name does not matter.
* d and e have the same structure as S and T (an int followed by a double), making them equivalent to a, b, and c.

Thus, under loose name (structural) equivalence, all variables a, b, c, d, and e would be considered type equivalent because they share the same structural definition, irrespective of the names S and T.

Problem 2

1. implies that and are both of type int.
2. can be curried to , meaning takes an and returns a fuction that takes an and returns a .
3. Since and are , and takes two values to return a , , will produce a double.
4. So, takes two values (through ) and produces a (through ), which can be represented as .