Part 1:

Eden Abadi

Noam Weiler

1. a. {T1 = [T3 -> T4], T5 = [T3 -> T4], T2=Number}
b. There is no MGU -> T1 = Number & T1 = Symbol - > contradiction.
c. {T1=T2}
d. {}.

2. In our implementation of the interpreter, we had a problem when we wanted to compute the recursive values defined in letrec expressions. This problem happened because we had to assign a value to the recursive variable, and in order to compute the value we had to use it. However, we don't have this problem in our type checker and that is because when computing the type of the recursive variable, we can use type variables to infer the type of the recursive variable and therefore we don't need to use its computed value as our interpreter before.

3. (define type_function
(lambda (x)
(lambda (y)
(lambda (z)
(lambda(t) t) z) y) x))

Test:

Const type = L5typeOf('(lambda (x) (lambda (y) (lambda (z) (lambda(t) t) z) y) x))');

Assert(isTvar(type)&&isTvar(unbox(type.contents))&&isTvar(unbox(unbox(type.contents))&&isTvar(unbox(unbox(unbox(type.contents))));