**Part 1:**

**Eden Abadi**

**Noam Weiler**

1. a. {T1 = [T3 -> T4], T5 = [T3 -> T4], T2=Number}

b. There is not MGU -> T1 = Number & T1 = Symbol - > contradiction.

c. {T1=T2}

d. Number – there is no need for MGU – it is already an MGU so {}.

**2**. we can do so because of a few reasons od differences between the interpreter and our type checker:

**a.** The type checker sees only program text, whereas the interpreter runs over actual data.

**b.** The type environment binds identifiers to types, whereas the interpreter’s environment binds identifiers to values or locations (boxes) - hence there is no need for recursive environment because there are no un computed values.

**c.**The type checker compresses (even infinite) sets of values into types, whereas the interpreter treats the elements of these sets distinctly.

**d.**The type checker always terminates – (therefore we don’t have to worry about infinite recursion), whereas the interpreter might not.

**e.** The type checker passes over the body of each expression only once, whereas the interpreter might pass over each body anywhere from zero to infinite times.

**3.** (lambda (x) (lambda(x) (lambda(x) (lambda(x) x))))