1   Lexical and Dynamic Scope

Consider the following program:

```
x = 161
def add(y)
   return x + y
def f(x)
   return add(3)
```

Recall that a lexically scoped variable is looked up according to the textual arrangement of its definition. By contrast, a dynamically scoped variable is looked up in the call stack that is active at the moment when a name needs to be resolved and found in the most recent stack frame.

1. What does `f(5)` return under lexically scoped rules? What about dynamically scoped rules?

2 Type Unification

1. In this problem, we will perform type inference in order to determine the type of the following `foldl` program.

   ```
   def foldl f z l = if l ≡ [] then z else foldl f (f z (head l)) (tail l)
   ```

   (a) Derive type constraints for the above program.

   (b) Based on the type constraint equations from the previous part, use the unification algorithm to derive the type of the entire program.

   (c) Suppose we want to count the number of `a`'s in a string. Let the string have type *List* Char. Write an appropriate function involving `foldl` that computes this quantity.

2. What are the types of the empty lists in the following expression?

   ```
   if x = [] then [] else x::y fi
   ```