Problem I1: Types - Easy

20 points

Problem My Submissions

Summary

You are developing a type system for a simple programming language, called *Lang*. Here is an example program:

```
(type even (int -> bool))
(type odd  (int -> bool))

(func (even n) (if (= n 0) true  (odd
(- n 1))))
(func (odd n)  (if (= n 0) false (even
(- n 1))))

(type fib (int -> int))
(type helper (int int int -> int))

(func (fib n) (helper 0 1 (+ n 1)))
(func (helper n a b) (if (= n 0) a
(helper (- n 1) a (+ a b))))
```

Lang has the following types:

- `int`, representing the integers (..., `-1`, `0`, `1`, `2`, etc),
- `bool` for booleans (`true` and `false`).
- `(P1 P2 ... PN -> R)` for any types `P1`, `P2`, ..., `PN` and `R`. Representing a function taking `N` arguments, and returning a value of type `R`.

A *Lang* program consists of a sequence of statements, each on their own line (Empty lines are ignored). A statement can be one of the following:

- A type declaration, of the form `(type NAME TYPE)` which introduces a new variable, with the given name and type.
- A function definition, of the form `(func (NAME V1 V2 ... VN) BODY)` which supplies the implementation of a function

names are given by `V1`, `V2`, ..., `VN` and the type of its body will match the return type in its declaration too.

Expressions in *Lang* are one of the following:

- An integer literal, like `-1`, `0`, `1`, `2` etc.
- A variable name, which can refer to a parameter of the function the expression is defined in, or any function declared before the expression in the program.
- A conditional of the form `(if c a b)` which evaluates to `a` if `c` evaluates to `true` and evaluates to `b` if `c` evaluates to `false`.
- A function call of the form `(f e1 e2 ... en)` where `f` is a variable name and `ei` is an expression for `1 <= i <= n`.

Problem

Initially, we want to know **how many types** are mentioned in the program. When counting mentions of compound types, we count the compound type but also its constituents sub-types, so the type `(int bool -> (-> int))` mentions **5 types**: `(int bool -> (-> int))`, `int`, `bool`, `(-> int)`, `int`. Note that we **count duplicates** as `int` is mentioned twice and counted both times.

Your input will be a *Lang* program with `N` lines. Count the total number of types mentioned according to this definition.

Note that *Lang* has the following built-in identifiers:

```
(type true  bool)
(type false bool)
(type +     (int int -> int))
(type -     (int int -> int))
(type =     (int int -> bool))
(type <     (int int -> bool))
```

Constraints

`1 <= N <= 100`

Example 1

`(int -> bool)` **twice**, and `(int int -> int)` and `(int int int -> int)` **once each**. So given the example program as input, the expected output would be:

Output

```
14
```

Sample Input

```
(type even (int -> bool)
(type odd  (int -> bool)

(func (even n) (if (= n 
(func (odd n)  (if (= n 

(type fib (int -> int))
(type helper (int int in

(func (fib n) (helper 0 
(func (helper n a b) (if
```

Sample Output

```
14
```