Report – Liquid Type Checking

For the project's phase 5, I decided to implement some liquid type checking in my plush compiler. This implementation has many limitations so I'm going to describe what is currently supported by the compiler. Currently it just works for the following types: int, boolean, float. You can make transitive variable declarations (Figure 1). You can declare the parameters of a function or procedure as liquid types as well as the function return type (not supported for FFI's). It's possible to declare variables and assign raw ints, floats, or booleans, other variables of those types and function calls. It's also possible to reassign these variables as raw ints, floats, booleans or other variables of these types. Another possible action is to typecheck a sum of variables, integers, or a mix of them, but this feature is only supported for binary sum operations and only works if the variables that are being summed aren't the parameters of a function/procedure.

```
var n1 : {int | n1 ⩾ 10} := 12;

var n2 : {int | n2 ⩾ 11} := n1;

var n3 : {int | n3 ⩾ 12} := n2;
```

Figure 1

There were some challenges namely:

1. Refactoring a substantial part of the type checking so that it works with and without liquid type checking. (overcame)
2. Understanding how to implement the transitive variable declarations and the type checking for the arguments of the function calls. (overcame)
3. Implementing the add operation inside function definitions. (it wasn't possible to implement due to time limitations).