Project Milestone 2 part A

Abhi Agarwal

My project two documentation is organized in a way where I break down the details I'm going to present and explain them in detail, and then I construct the table.

2.1 - Assignment compatible

Definition: Assignment compatible applies to us here because we know that one type of a value (T_1) is assignment compatible with a second type of a value (T_2) if the value of the first type can be assigned to the value of the second type (if $type_2\ T_2 = T_1$ is possible).
Analysis: To me (my assumption is that) `any` means that we're able to convert between types very explicitly without using a cast, and we're able to assign certain values regardless of the type. Moreover, the every type is assignment compatible with itself is fairly straight forward, and the standard case.

2.2 - Literals

Identifier

Definition: Identifier are tokens that start with a letter, $, or _, followed by more of the same as well as digits, except that keywords (literal tokens used by the grammar) are not permitted as identifiers.
Scope: Here we're looking at values within the scope we're already in.
Analysis: Identifier literals must be declared within the scope where it is declared, and has to have the correct type. We have to do a type check to see if the assignment is the same as the type it was declared as.

Integer

Definition: Integer tokens are sequences of digits.
Scope: Here we're looking at values within the scope we're already in.
Analysis: We've to check if they've the type int to classify them as Integers. This is just mechanical as we've to simply check if their type matches the type `int`.

String

Definition: String tokens are sequences of characters enclosed in either ' (single quote) or " (double quote).
Scope: Here we're looking at values within the scope we're already in.
Analysis: We've to check if they've the type string to classify them as Strings.

object

Definition: object literal $\{k_1$:Literal1,... ,k_n:Literaln$\}$ for n $\geq$ 1 with each key k_i an Identifier (also called "field name").
Scope: Here we've to create a new scope as m_k must be a distinct Identifier, and it must be distinct in the sense that they are unique to all other $(m_1, m_2, ..., m_k)$ Identifiers within the enclosing of the curly braces – $\{\}$.
Analysis: First we've to check if m_k for a given k is not already exists in the scope we're in, if it does then we've to generate an issue or an error for the user as we're strictly not allowing this in the language. We also have to check for the type of l_k and if it has a type which is assignment compatible to the type declared for the member.

2.3 - l-value