MyPL Type Inference Rules. The goal of the following rules are to help clarify type inference and type checking in `MyPL`. We make the following assumptions.

- The notation $e : t$ says that expression e has type t.
- The **`null`** value has type **`void`** (i.e., **`null`** : **`void`**) and literals (constants) c have their corresponding type t (i.e., $c : t$).
- In general, e denotes an expression, t a data type, x a variable/field name, s a struct name, and f a function name.
- Function types are denoted as mappings from parameter type lists (in the order of function parameters) to return types. For example, $f : \texttt{int}, \texttt{string} \rightarrow \texttt{char}$ says f takes an int and string, and returns a character.
- Struct types are denoted as dictionaries mapping field names to types.
- Function parameters and struct fields are assumed not to have function names as types.
- Array types are represented using square brackets, e.g., an int array is denoted as [**`int`**].
- Γ is the typing context (environment). The notation $\Gamma \vdash e : t$ says that the current context implies that expression e has type t. Similarly, the notation $\Gamma,\ \text{stmt} \vdash e : t$ says that the current context extended with the statement implies e has type t. We take some liberties below by assuming we are "in" the statement (stmt) when it extends the scope.
- Unlike syntax rules, the typing rules are meant to provide a guide to some of the details as opposed to an implementation strategy.

Typing Rules for MyPL Expressions:

$$\frac{\Gamma \vdash e_1 : t \quad \Gamma \vdash e_2 : t \quad t \in \{\texttt{int}, \texttt{double}\} \quad \texttt{op} \in \{\texttt{+}, \texttt{-}, \texttt{*}, \backslash\}}{\Gamma \vdash e_1 \ \texttt{op}\ e_2 : t} \tag{1}$$

$$\frac{\Gamma \vdash e_1 : t_1 \quad \Gamma \vdash e_2 : t_2 \quad (t_1 = t_2 \ \vee\ t_1 = \texttt{void} \ \vee\ t_2 = \texttt{void}) \quad \texttt{op} \in \{\texttt{==}, \texttt{!=}\}}{\Gamma \vdash: e_1 \ \texttt{op}\ e_2 : \texttt{bool}} \tag{2}$$

$$\frac{\Gamma \vdash e_1 : t \quad \Gamma \vdash e_2 : t \quad t \in \{\texttt{int}, \texttt{double}, \texttt{char}, \texttt{string}\} \quad \texttt{op} \in \{\texttt{<}, \texttt{>}, \texttt{<=}, \texttt{>=}\}}{\Gamma \vdash: e_1 \ \texttt{op}\ e_2 : \texttt{bool}} \tag{3}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1 \ \texttt{and}\ e_2 : \texttt{bool}} \tag{4}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1 \ \texttt{or}\ e_2 : \texttt{bool}} \tag{5}$$

$$\frac{\Gamma \vdash e : \texttt{bool}}{\Gamma \vdash \texttt{not}\ e : \texttt{bool}} \tag{6}$$

Typing Rules for MyPL Statements:

$$\frac{\Gamma \vdash e : t' \quad t' \in \{\texttt{void}, t\}}{\Gamma,\ t\ x = e\ \vdash x : t} \tag{7}$$

$$\frac{\Gamma \vdash x : t}{\Gamma,\ x = e\ \vdash e : t' \quad t' \in \{\texttt{void}, t\}} \tag{8}$$

$$\frac{}{\Gamma,\ \texttt{while (}e\texttt{)}\ \ldots\ \vdash e : \texttt{bool}} \tag{9}$$

$$\frac{\Gamma \vdash e : t' \quad t' \in \{\texttt{void}, t\}}{\Gamma,\ \texttt{for (}t\ x = e\texttt{;}\ \ldots\texttt{;}\ \ldots\texttt{)}\ \ldots\ \vdash x : t} \tag{10}$$

$$\frac{}{\Gamma,\ \texttt{for (}\ldots\texttt{;}\ e\texttt{;}\ \ldots\texttt{)}\ \ldots\ \vdash e : \texttt{bool}} \tag{11}$$

$$\frac{\Gamma,\ \texttt{for (}\ldots\texttt{;}\ \ldots\texttt{;}\ x = e\texttt{)}\ \ldots\ \vdash x : t}{\Gamma,\ \texttt{for (}\ldots\texttt{;}\ \ldots\texttt{;}\ x = e\texttt{)}\ \ldots\ \vdash e : t' \quad t' \in \{\texttt{void}, t\}} \tag{12}$$

$$\frac{}{\Gamma, \texttt{if (}e\texttt{)}\ \ldots\ \vdash e : \texttt{bool}} \tag{13}$$

$$\frac{}{\Gamma,\ \ldots\ \texttt{elseif (}e\texttt{)}\ \ldots\ \vdash e : \texttt{bool}} \tag{14}$$

Typing Rules for MyPL Structs:

$$\frac{t_i \neq \texttt{void}}{\Gamma,\ \texttt{struct}\ s\ \{t_1\ x_1,\ \ldots\ , t_n\ x_n\ \}\ \vdash s : \{x_1 \to t_1,\ \ldots\ , x_n \to t_n\}} \tag{15}$$

$$\frac{\Gamma \vdash e : s \quad \Gamma \vdash s : \{\ldots\ ,\ x_i \to t_i,\ \ldots\}}{\Gamma \vdash e\texttt{.}x_i : t'_i \quad t'_i \in \{\texttt{void}, t_i\}} \tag{16}$$

$$\frac{\Gamma \vdash s : \{x_1 \to t_1, \ldots, x_n \to t_n\}}{\Gamma \vdash \texttt{new}\ s : s} \tag{17}$$

Typing Rules for MyPL Functions:

$$\frac{t_i \neq \texttt{void}}{\Gamma,\ t\ f\texttt{(}t_1\ x_1,\ \ldots,\ t_n\ x_n\texttt{)\{}\ \ldots\ \texttt{\}}\ \vdash f : t_1,\ \ldots, t_n \to t} \tag{18}$$

$$\frac{\Gamma \vdash f : t_1,\ \ldots,\ t_n \to t \quad \Gamma \vdash e_i : t_i' \quad t_i' \in \{\texttt{void}, t_i\}}{\Gamma \vdash f(e_1, \ldots, e_n) : t} \tag{19}$$

$$\frac{\Gamma \vdash \textit{return} : t}{\Gamma,\ \texttt{return}\ e\ \vdash e : t' \quad t' \in \{\texttt{void}, t\}} \quad \dagger \tag{20}$$

Typing Rules for MyPL Arrays:

$$\frac{\Gamma \vdash e : t' \quad t' \in \{\texttt{void}, [t]\}}{\Gamma,\ \texttt{array}\ t\ x = e\ \vdash\ x : [t]} \tag{21}$$

$$\frac{\Gamma \vdash x : [t]}{\Gamma,\ x = e\ \vdash\ e : t' \quad t' \in \{\texttt{void}, [t]\}} \quad \ddagger \tag{22}$$

$$\frac{\Gamma \vdash e : \texttt{int} \quad t \notin \{\texttt{void}, [t']\}}{\Gamma \vdash\ \texttt{new}\ t\texttt{[}e\texttt{]} : [t]} \tag{23}$$

$$\frac{\Gamma \vdash e_1 : [t] \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1\texttt{[}e_2\texttt{]} : t} \tag{24}$$

$$\frac{t_i \notin \{\texttt{void}, [t']\}}{\Gamma,\ \texttt{struct}\ s\ \texttt{\{}\ldots,\ \texttt{array}\ t_i\ x_i,\ \ldots\texttt{\}} \vdash s : \{\ldots,\ [t_i],\ \ldots\}} \tag{25}$$

$$\frac{t_i \notin \{\texttt{void}, [t']\}}{\Gamma,\ t\ f\texttt{(}\ldots,\ \texttt{array}\ t_i\ x_i,\ \ldots\texttt{)} \vdash f :\ \ldots,\ [t_i],\ \cdots \to t} \tag{26}$$

$$\frac{t \notin \{\texttt{void}, [t']\}}{\Gamma,\ \texttt{array}\ t\ f\texttt{(}\ \ldots\ \texttt{)} \vdash f :\ \ldots\ \to [t]} \tag{27}$$

† where "*return*" is a special variable assumed in each function context with the corresponding return type

‡ This rule is reduntant with rule 8