This is the language specification for a specific syntax of system F (augmented with additional constructs) used by this compiler

l	:=	**null**	unit literal
		true | **false**	boolean literals
		... | **~2** | **~1** | **0** | **1** | **2** | ...	64 bit signed integers
e	:=	l	literals
		id	identifier
		(e **)**	parenthesized
		e_1 e_2	application
		e_1 op e_2	infix operation
		e [<τ>$^{(,)}$]	type application
		forall <id>$^{(,)}$**.** e	parametric polymorphism
		if e_1 **then** e_2 **else** e_3	if expression
		lambda <id **:** τ>$^{(,)}$**.** e	anonymous function
		let vid **:** τ **=** e_1 **in** e_2 **end**	
d	:=	**val** vid **:** typ **=** exp	value bind
τ	:=	tid	type identifier
		(τ **)**	parenthesized
		<τ>$^{(*)}$	tuple types
		τ_1 **->** τ_2	arrow types
		univ tid**.** τ	universal types
		int | **bool** | **unit**	primitive types

Supported built-in types: **int**, **bool**, **unit**.

Not supporting declaration of operators as infix.

No identifiers can be rebound.

fn arguments must be surrounded by parentheses, and must be a (potentially empty) comma-separated list of value identifiers. In particular, there are only single-argument functions.

fn arguments need to be type annotated.

Type identifiers that start with **'**, such as **'a**, are polymorphic. Type identifiers that start with **''** are equality types.