Q Language

# Scoping Rules

## Levels of Scope

* Block Scope: Definitions inside a block are valid only for that block and any children. This includes things like if and while statements.
* Function Scope: Functions have local variables. These are only valid for that function. Functions do not have inner functions in Q so we do not need to worry about this.

## Restrictions

* Re-declaring variables in inner scopes is valid (like in javascript/java).
* Static scoping

## Errors

* If a variable is accessed, but cannot be found in the scope, then we throw an error (Undeclared Variable). This could also be because the variable is defined after it is accessed (although Q encourages definitions to be before any code).
* Variable already declared in current scope.

## Notes to programmer

* Be careful when coding function definition scopes. The parameter and the body must both be considered.

# Type Checking

## Rules/Errors

Our language will carry out some implicit conversion where convenient. Int and Float will both be of type “Number”.

## Variables

* Any two numbers can arithmetic operations carried out on them. If their types differ, we will convert both to Float.
* We will not offer any C-style bitwise operations on primitives. All types are abstract and do not necessarily correspond exactly to addresses in memory.
* In assignment, the type of the right hand side must match the type of the variable (unless we can convert the RHS to a float).
* Variables can take assignment from the output of a function. We check the function return type is the same type as the variable

## Functions

* When checking a function call for type errors, we must remember to check that the parameters in the call match the types of the parameters in the definition.
* Return types of functions must be respected. We must also ensure that a return statement is specified if the function is not void

## Comparisons

* We allow comparisons between integers, floats and characters.