Charles Walker
5/1/21

MYPL REPL Project Final Report:

The MYPL REPL can take user input (statements), evaluate any expression and display it to the user. This is done in the REPL environment as opposed to a mypl program file. Here is what I accomplished

In the Repl session:

- A return statement consisting of only an ID followed by a COLON is considered a REPL endpoint, and the value of the ID will be evaluated and displayed to the user.
- When an expression is entered that would otherwise be considered an unfinished statement, it is considered a REPL endpoint and will be evaluated and displayed to the user
- A REPL endpoint is an expression that is evaluated and displayed after the user presses enter.
- After entering a REPL endpoint, the loop in the MYPL driver (hw6.cpp) will continue to create REPL nodes until EOF is found (ctrl+d)

For the REPL to work, the following changes were made:

In hw6.cpp:

- Create the Repl Nodes
- Loop through those nodes until EOF detected
    - parse through the node
    - type check the node
    - interpret the node

In Parser:

- A Repl Endpoint is a return statement in a REPL session
    - Added a boolean "in_repl" to distinguish whether a return statement is an endpoint or not
- REPL node function:
    - Loop through statements
        - If the statement is normal, push back the statement to the AST REPL node
        - If the statement is a repl endpoint, stop
- A public boolean variable that tells when EOF is found

In AST:

- REPL node:
    - List of statements
    - Possibly list of decls

In Type_Checker:

- TypeChecker::visit(Repl& node)

    - Behaves similarly to the Program& node but loops through the Repl statements

In Interpreter:

- Interpreter::visit(Repl& node)
    - Is set up similarly to Program& node but instead loops through the REPL statements.

Not Completed:

The REPL was supposed to accommodate function and type declarations in the REPL environment, but this was not achieved.