Written Homework.

Read the following in the textbook and then answer the questions below.

- Ch. 11: pg. 333–334

(1). Describe what the book means when it refers to “state” and the significance of state when it comes to programming languages.

The following questions require you to use GHC, which you can either install on your own machine (recommended) or use the version installed on ada. GHC can be downloaded from http://www.haskell.org, which also has general information and documentation about Haskell.

(2). Start ghci (e.g., by running the ghci command on ada), guess what you think each command should evaluate to, and then test your guess (by typing the expression directly into the interpreter). Write down your guess and what was actually returned. Note that some may return errors.

(a). 3 * 5 + 8
(b). (+) ((*) 3 5) 8
(c). 2^1000
(d). even 9
(e). succ 6
(f). succ (pred 6)
(g). gcd 21 27
(h). gcd(21, 27)
(i). [1..4]
(j). [1..4] ++ [5..10]
(k). 1 : [2,4 .. 10]
(l). [1, "2"]

(3). List the types for each valid expression from (2), i.e., each expression that didn’t result in an error. You can get the type of an expression by using the :type command in the interpreter, e.g.,

```
Prelude> :type 42
42 :: Num a => a
```

In this case the type of 42 is “Num a => a”.

1
(4). Bind the identifier `zp` to an infinite list of positive integers using the “let \( v = e \)” syntax:

```
Prelude> let zp = [1..]
```

Based on `zp`, perform the following (via the interpreter) and describe what you think is going on with each example.

(a). What is the type of `zp`?

(b). What is the result of executing:

```
Prelude> head zp
```

(c). What is the result of executing:

```
Prelude> head (tail zp)
```

(d). What is the result of executing:

```
Prelude> head (drop 100 zp)
```

(e). What is the result of executing:

```
Prelude> take 20 (filter even zp)
```

(f). What is the result of evaluating the entire list:

```
Prelude> zp
```

Note that to stop printing type Ctrl-c (i.e., hold the control key and type the letter c)

**Programming Homework: Basic Interpreter.**

The goal of this assignment is to implement a basic interpreter for non-function and non-struct statements in MyPL. Thus, your interpreter should handle all variable declarations, assignment statements, while statements, if statements, and expression statements (including calls to built in functions) that do not involve struct object creation and user-defined function calls. Your job is to implement a new visitor class that performs interpretation over a type-checked AST structure generated from your parser.

On the due date, **hand in** a cover sheet together with hard copy of your implementation (all new or edited source code files), print outs of tests showing your program works properly, a write up of your testing strategies and implementation issues, and your test cases (test files). In addition, submit all of the source files needed to execute your program as well as your test cases to the online dropoff site (https://www.cs.gonzaga.edu/dropoff/).

As part of your implementation you must use the code provided program design (which is just the visitor class and `hw6.py` file). Also be sure to add comments to your code! Within your classes, you can define any helper functions you see fit. Note that you will be re-using the symbol table from HW5 as well to store variable value information (i.e., the value of identifiers as opposed to the type of identifiers).
hw6.py

#!/usr/bin/python3
#
# Author:
# Assignment: 6
# Description:
#   Simple script to execute the MyPL interpreter.
#---------------------------------------------------------------------------

import mypl_error as error
import mypl_lexer as lexer
import mypl_token as token
import mypl_parser as parser
import mypl_ast as ast
import mypl_type_checker as type_checker
import mypl_interpreter as interpreter
import sys

def main(filename):
    try:
        file_stream = open(filename, 'r')
        hw6(file_stream)
        file_stream.close()
    except FileNotFoundError:
        sys.exit('invalid filename %s' % filename)
    except error.MyPLError as e:
        file_stream.close()
        sys.exit(e)

def hw6(file_stream):
    the_lexer = lexer.Lexer(file_stream)
    the_parser = parser.Parser(the_lexer)
    stmt_list = the_parser.parse()
    the_type_checker = type_checker.TypeChecker()
    stmt_list.accept(the_type_checker)
    the_interpreter = interpreter.Interpreter()
    stmt_list.accept(the_interpreter)

if __name__ == '__main__':
    if len(sys.argv) != 2:
        sys.exit('Usage: %s file' % sys.argv[0])
    main(sys.argv[1])
class Interpreter(ast.Visitor):
    """A MyPL interpret visitor implementation"""

    def __init__(self):
        # initialize the symbol table (for ids -> values)
        self.sym_table = sym_tbl.SymbolTable()
        # holds the type of last expression type
        self.current_value = None

        def __error(self, msg, the_token):
            raise error.MyPLLError(msg, the_token.line, the_token.column)

        def visit_stmt_list(self, stmt_list):
            self.sym_table.push_environment()
            for stmt in stmt_list.stmts:
                stmt.accept(self)
            self.sym_table.pop_environment()

        ... etc ...