Written Homework.

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

- Ch. 5.1–5.3, 5.5.1

(1). This question has two parts.

(a). Briefly compare our approach for dealing with function calls via the symbol table to the idea of “activation records” from the textbook. In particular, describe how the various elements of an activation record for functions in the book are mapped to our MyPL implementation.

(b). Consider the following MyPL program.

```plaintext
1: fun int fac(n: int)
2:     if n <= 0 then
3:         return 1;
4:     else
5:         var r = n * fac(n - 1);
6:     return r;
7: end
8: end
9: var v = 3;
10: fac(v);
```

Using our approach for implementing function calls (as opposed to the textbook’s), draw the symbol table for each call to the `fac` function. In particular, draw the symbol table prior to the execution of line 10, then the symbol table for the first execution (the `fac(3)` call), then the symbol table for the second execution (the `fac(2)` call), then the symbol table for the third execution (the `fac(1)` call), and so on. For example, the first execution will look something like this (where the stack grows down the page and “?” denotes the return value from the next function call):

```
+-------------------+
| fac - [global,...] | <= global
| v - 3              |
+-------------------+
| n - 3              | <= fac(3)
| r - 3 * ?          |
+-------------------+ top - +-------------------+
```

The following questions are related specifically to define basic functions in Haskell. Print your definitions for each function below as well as your tests showing that your definitions work correctly. Note you will need more test cases than just those given below. Finally, note that you are only allowed to use an `if-then-else` expression for question (3).

(2). Write a function `unordered3` that returns true if its three arguments are not in ascending or descending order (i.e., not ordered) and false otherwise. E.g., `unordered3 1 3 2` and `unordered3`
1 2 1 should return true, whereas unordered3 1 2 3, unordered3 3 2 1 and unordered3 1 1 1 should all return false.

(3). Write a function quadrant that takes a point and returns the corresponding quadrant where the point is located (see en.wikipedia.org/wiki/Quadrant_(plane_geometry)). For example, quadrant (3, 5) should return 1, whereas quadrant (-3, 5) should return 2. If the x or y value of the point is zero, then quadrant (x, y) should return 0.

(4). Write a function distance that returns the distance between two points. E.g., distance (0, 0) (2, 0) computes the distance between points (0, 0) and (2, 0), which is 2.0, and distance (-2, 1) (1, 5) computes the distance between points (-2, 1) and (1, 5), which is 5.0. Hint: the sqrt function included with Haskell returns the square root.

(5). Assume a date is represented using a 3-tuple (m, d, y) where m is the month, d is the day, and y is the year. Write a function after that takes two dates and returns true if the first date occurs after the second date. For example, after (4, 3, 2000) (3, 4, 2000) should return true (since the first date represents April 3, 2000 and the second date represents March 4, 2000).

Programming Homework: Completing the Interpreter.

The goal of this assignment is to finish your interpreter for supporting functions and structs in MyPL. In particular, your job is to finish the remaining functions in your visitor class from HW 6. As part of this, you will need to use the extended symbol table (given below) as well as add a new type-checking constraint for variable shadowing. The extended symbol table is given below, and the general strategy for the homework assignment and additional static constraint is explained in the lecture notes.

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 finishing your implementation you must use the code provided below (which is just the extended symbol table and hw7.py file). Note that within your classes, you can define any helper functions you see fit.
hw6.py

#!/usr/bin/python3
#
# Author:
# Assignment: 7
# Description:
# Simple script to execute the MyPL interpreter.
#--------------------------------------------------------

import mypl_error as error
import mypl_lex 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 hw7(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()
    the_interpreter.run(stmt_list)

if __name__ == '__main__':
    if len(sys.argv) != 2:
        sys.exit('Usage: %s file' % sys.argv[0])
    main(sys.argv[1])
mypl_symbol_table.py

class SymbolTable(object):
    """A symbol table consists of a stack of environments, where each
    environment maps a (variable) name to its associated information
    """

def __init__(self):
    self.scopes = []    # list of {id_name:info}
    self.env_id = None  # current environment in use

def __get_env_index(self):
    for i, scope in enumerate(self.scopes):
        if self.env_id == id(scope):
            return i

def __environment(self, name):
    # search from last (most recent) to first environment
    index = self.__get_env_index()
    for i in range(index, -1, -1):
        if name in self.scopes[i]:
            return self.scopes[i]

def id_exists(self, identifier):
    return self.__environment(identifier) != None

def id_exists_in_env(self, identifier, env_id):
    for scope in self.scopes:
        if env_id == id(scope):
            return identifier in scope

def add_id(self, identifier):
    # can't add if no environments
    if not self.scopes:
        return
    # add to the current environment id
    self.scopes[self.__get_env_index()][identifier] = None

def get_info(self, identifier):
    env = self.__environment(identifier)
    if env is not None:
        return env[identifier]

def set_info(self, identifier, info):
    env = self.__environment(identifier)
    if env is not None:
        env[identifier] = info
def push_environment(self):
    new_scope = {}
    if len(self.scopes) == 0:
        self.scopes.append(new_scope)
    else:
        index = self.__get_env_index()
        if index == len(self.scopes) - 1:
            self.scopes.append(new_scope)
        else:
            self.scopes.insert(index + 1, new_scope)
    self.env_id = id(new_scope)

def get_env_id(self):
    return self.env_id

def set_env_id(self, env_id):
    self.env_id = env_id

def pop_environment(self):
    if len(self.scopes) <= 0:
        return
    index = self.__get_env_index()
    del self.scopes[index]
    if index > 0:
        self.env_id = id(self.scopes[index - 1])
    else:
        self.env_id = None

def __str__(self):
    s = ''
    for i, scope in enumerate(self.scopes):
        s += '*' + str(id(scope)) + ': ' + str(scope) + '\n'
    return s
import mypl_token as token
import mypl_ast as ast
import mypl_error as error
import mypl_symbol_table as sym_tbl

class ReturnException(Exception): pass

class Interpreter(ast.Visitor):
    """A MyPL AST interpreter 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
        # the heap {oid:struct_obj}
        self.heap = {}

        ... rest of visitor functions ...