Name: Armando Valdez

Assignment: HW7 **Date:** 4/7/20

Program Submission. To receive a grade, you must:

- Electronically submit all source code needed to run and test your program
- Turn in hard-copy with a cover sheet (this page), test cases, example program runs, and a discussion
- Ensure your source code compiles without errors or warnings

Test Cases (5 Points). You must develop test cases beyond those given. You will be evaluated on:

- The breadth and quality of the test cases you develop
- One paragraph write-up describing the rationale behind your choice of test cases

Complete & Correct (20 Points). The majority of your grade is based on completeness and correctness:

- Code does the requested tasks using the requested techniques
- Code runs without runtime errors

Format (2 Points). Your code must be formatted professionally and will be graded on:

- Consistent indentation, white space, variable names, etc.
- Consistently follow good programming style for language used

Comments (2 Points). Your code must be documented appropriately using comments, including:

- File headings with author name, assignment number, date, and description (purpose and any issues)
- Meaningful variable, class, and function names
- Meaningful comments of difficult and/or complex parts of code
- Meaningful function comments

Discussion (6 Points). You must include a discussion and reflection of your work:

- One paragraph description of any challenges and/or issues and how addressed
- One paragraph reflection on process and/or code improvements

Total Score: ______ / 35

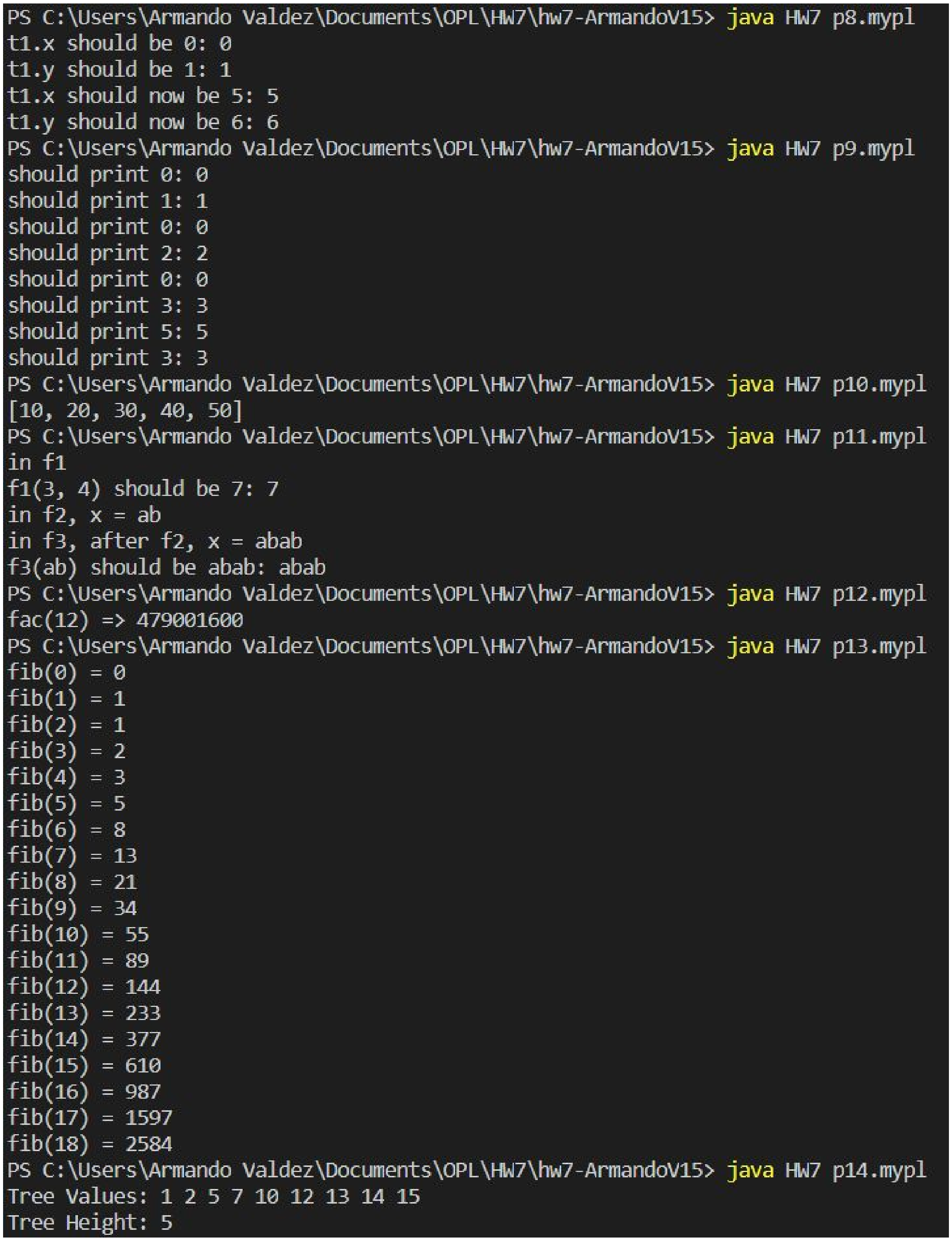
Test cases

```
# Performing arithmetic using user-defined types

type Type1
    var x := 5
    var y := 8
end

var t1 := new Type1
var sum := t1.x + t1.y
set t1.x := 20
set t1.y := 10
var div := t1.x / t1.y
print(itos(sum))
print("\n")
print(itos(div))
print("\n")

# Performing arithmetic in a simple linked list

type Node
  var val := 0
  var Node next := nil
end

var head := new Node
set head.val := 10

var len := 6
var i := 1
var ptr := head
```

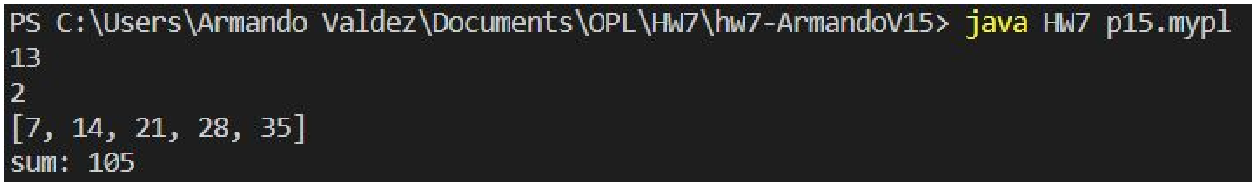
```
while i < len do
  set ptr.val := i * 7
  set ptr.next := new Node
  set ptr := ptr.next
  set i := i + 1
end

print("[")
set i := 1
set ptr := head
while i < len do
  print(itos(ptr.val))
  if i < (len - 1) then
    print(", ")
  end
  set ptr := ptr.next
  set i := i + 1
end
print("]\n")

var add := 0

set add := add + head.val
set head := head.next
set add := add + head.val
set head := head.next
set add := add + head.val
set head := head.next
set add := add + head.val
set head := head.next
set add := add + head.val
set head := head.next
```

```
print("sum: ")
print(itos(add))
```

Example Runs



```
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p8.mypl
t1.x should be 0: 0
t1.y should be 1: 1
t1.x should now be 5: 5
t1.y should now be 6: 6
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p9.mypl
should print 0: 0
should print 1: 1
should print 0: 0
should print 2: 2
should print 0: 0
should print 3: 3
should print 5: 5
should print 3: 3
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p10.mypl
[10, 20, 30, 40, 50]
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p11.mypl
in f1
f1(3, 4) should be 7: 7
in f2, x = ab
in f3, after f2, x = abab
f3(ab) should be abab: abab
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p12.mypl
fac(12) => 479001600
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p13.mypl
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
fib(6) = 8
fib(7) = 13
fib(8) = 21
fib(9) = 34
fib(10) = 55
fib(11) = 89
fib(12) = 144
fib(13) = 233
fib(14) = 377
fib(15) = 610
fib(16) = 987
fib(17) = 1597
fib(18) = 2584
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p14.mypl
Tree Values: 1 2 5 7 10 12 13 14 15
Tree Height: 5
```



```
PS C:\Users\Armando Valdez\Documents\OPL\HW7\hw7-ArmandoV15> java HW7 p15.mypl
13
2
[7, 14, 21, 28, 35]
sum: 105
```

Test case reflection

As mentioned on piazza a lot of the test cases provided for this assignment covered a lot of what could be tested with the implementation of type declarations and function declarations, but one aspect I didn't really notice was arithmetic being used in incorporation with types and linked lists. For my test cases I showed how arithmetic can be incorporated by adding up an already created linked list to get the sum of the entire list, as well as performing some sort of operation on defined type values then manipulating those values and performing another operation on them to show that they are being changed.

Discussion

For me the main issue I had was completing IDRValue and LValue. Coming over from HW6 I didn't include a todo to modify them so that they accept paths of lengths greater than one so when I tested my code for the first time I was getting errors everywhere. It wasn't until I reached out for help that I realized I needed to modify these two functions. While modifying these two functions I realized they were very similar in structure but one thing I got stuck on was incorporating a heap into both of them but after doing some research on this process I was able to fix my problem and finish the two functions.

The process for completing this code was very straight forward for the most part. There wasn't much leftover from HW6 to complete but the functions that were left definitely did take some time to work through and figure out. One place where I believe there could be some improvements in my code would have to be my return functions in the interpreter. It works correctly for all test cases but I feel like I could improve the way it handles returns in the future.