Homework 3 Report

Pınar Yücel

CS 315-3

21802188

Code

```
# Nested Subprogram Definitons
println("Nested Subprogram Definitions")

# I am trying to write a very simple nested function [1].
function sumTimesConstant(x,y)
   function helper1(a,b)
      return helper2(a + b)
   end

   function helper2(c)
      return c * 5
   end

   return helper1(x,y)
end

# If it prints 35 then the previous definition for a nested subprogram is working [1].
println(sumTimesConstant(3,4))

# Scope of Local Variables
println("\nScope of Local Variables")

# I am testing what happens to the scope of a local variable when defined inside function and used inside a function defined inside that function [2].
function printA()
   a = 5;

   function printAHelper()
      println(a)
   end

   printAHelper()
end

# If it prints 5 then println sees the previously defined local variable [2].
printA()
println()
```

```
# I am testing what happens when there is already a local variable but a new one with the same name is created [2].
function printB()
   b = 5;

   function printBHelper()
      b = 4
      println(b)
   end

   printBHelper()
   println(b)
end

# If it prints two 4s then it means that the statement b = 4 does not create a new variable but is uses the existing one. If it first prints 4 then 5, then it means the staementb = 4 causes a new variable to be defined [2].
printB()
println()

# I am testing what happens when there is already a local variable but a new one with the same name is created with "local" [2].
function printC()
   c = 5;

   function printCHelper()
      local c = 4
      println(c)
   end

   printCHelper()
   println(c)
end

# If it prints 4 and then 5, it means that "local" word caused the creation of a new variable [2].
printC()
println()

# I am testing what happens when there is a global variable with the same name as a local variable inside a function [2].
```

```
d = 5

function printD()
   d = 4;

   println(d)
end

# If it prints 4 and then 5 then it means that d = 4 statement creates a new local variable [2].
printD()
println(d)
println()

# I am testing what happens when there is a global variable with the same name as a local variable inside a loop [2].
e = 5
counter = 0

while(counter < 1)
   e = 4
   println(e)
   counter = counter + 1
end

# If it prints 4 then it means the statement e = 4 inside the loop did not caused the creation of a new local variable [2].
println(e)
println()

# I am testing what happens when a variable with the same name is defined by local word in an outer loop inside a function [2].
function sum(var1, var2)
   sum = 0

   for n = 1:1
      local sum =  var1
      sum = sum + var2
   end

   println(sum)
```

```
end

# If it prints 0, then local sum is diffent than sum [2].
sum(1,2)
println()

# I am testing what happens when a variable is defined without local word in an outer loop inside a function [2].
function sum(var1, var2)
   sum = 0

   for n = 1:1
      sum =  var1
      sum = sum + var2
   end

   println(sum)
end

# If it prints 3, then all the sums above are the same variable [2].
sum(1,2)
println()

a = 1
b = 2
c = 3

# I am testing the let scope [2].
let b = 1, a = 2
   c = 1
   println(a)
   println(b)
   println(c)
   println()
end

# If it prints 1 and 2 and 1, then it means a and b defined in let are different variables. However c is not [2].
println(a)
println(b)
```

```
println(c)
println()

# Parameter Passing Methods
println("Parameter Passing Methods")

a = [1,2,3]

# I am trying to test what happens when I push an element into an array in a function [3].
function pushToArray(arr)
    push!(arr, 4)
end

pushToArray(a)

# If it prints 1 2 3 4, then it means the array was modified in the function [3].
println(a)
println()

a = [1,2,3]

# I am trying to test what happens when I try to change the array by re defining it [3].
function modifyArray(arr)
    arr = [4,5,6]
end

modifyArray(a)

# If it prints 1 2 3, then it means the array was not changed in the function [3].
println(a)
println()

a = 5

# I am trying to test what happens when I modfiy an integer inside a function [3].
function modifyInteger(int)
    int = 4
end

modifyInteger(a)
```

```
# If it prints 4, then it means the integer was modified inside the function. If it prints 5 then it means the integer was not modified in the function [3].
println(a)
println()

# Keyword and Default Parameters
println("Keyword and Default Parameters")

# I am trying to test how the keyword parameters work [3].
function printInOrder(;first = 3, second = 2, third = 1)
   println(first , " " , second , " " , third)
end

# If it prints 1 2 3, then it means keyword argument passing worked successfully [3].
printInOrder(third = 3, first = 1, second = 2)
println()

# I am trying to test default parameters [3].
function printColors(color1, color2 = "blue", color3 = "red")
   println(color1, " ", color2, " ", color3)
end

# If if prints orange blue red, then default parameters work as expected [3].
printColors("orange")

# If it prints orange green red then default parameters work as expected [3].
printColors("orange", "green")
println()

## Closures
println("Closures")

# I am testing the closures with a very simple function [4].
function topLevel(a)
   b = 2
   println(a)
   println(b)

   function closure(c)
```

```
        d = 4
        println(a)
        println(b)
        println(c)
        println(d)
    end
end

# If it prints 1 and 2, then it works as expected [4].
x = topLevel(1)
println()

# If it prints 1 to 4, then it works as expected [4].
x(3)
println()

# I am testing the same thing with another parameter. If it prints 0 and 2 then its working [4].
y = topLevel(0)
println()

# I am testing the same thing with another parameter. If it prints 0 2 4 4 then its working [4].
y(4)

# The outputs of x and y should be different as they are different closures [4].
```

Output

```
Nested Subprogram Definitions
35

Scope of Local Variables
5

4
4

4
5

4
```

5

4
4

0

3

2
1
1

1
2
1

Parameter Passing Methods
[1, 2, 3, 4]

[1, 2, 3]

5

Keyword and Default Parameters
1 2 3

orange blue red
orange green red

Closures
1
2

1
2
3
4

0

2

0
2
4
4

Evaluation

I think nested subprograms are not readable because you may have to follow the code carefully to understand which function calls which function with which parameter. Furthermore, since sub functions can go between the code in the outer function it makes nested subprograms even more unreadable. It would be hard to debug a program that uses nestes subprograms therefore its a minus in terms of writability because it makes it open to mistakes. I think since there are some rules that may cause confusion in scoping of local variables, these rules make the language less readable, such as the difference between the rules that apply loops and functions. Since their rules are different, it makes the language less writable because I can easily make mistakes by misremembering those rules. I also think that parameter passing methods are unreadable because it applies different rules to different kinds of data types and this is confusing and unwritable because I have to remember those rules and therefore I can easily make mistakes by misremembering. However, I think keyword parameters and default parameters are both readable because they are intuitive and writable because since they are intuitive I am less likely to make mistakes. In terms of closures, I think they are unreadable because you have to follow and remember the code carefully to understand it and also they do not look intuitive sometimes. I think they are also unwritable because since they are not intuitive, I can easily make mistakes writing them.

Learning Strategy

I used Google to learn the specified aspects of Julia. I tried to use the websites that are the either the offical website of Julia or known sites. I tried to avoid question ask sites such as stack overflow. I reported every experiment I performed as comments in the code. I did not continue experimenting without completely understanding an operation. I did not try anything random to see if it would work. I followed the information that I found on the internet. Since online compilers were allowed, I just tested my code on the online compiler whose link can be found below. According to those, outputs are as they should be.

Online Compiler

https://www.tutorialspoint.com/execute_julia_online.php

References

[1] “Nested functions and scope in Julia,” *The Craft of Coding*, 22-Sep-2020. [Online]. Available: https://craftofcoding.wordpress.com/2017/04/01/nested-functions-and-scope-in-julia/. [Accessed: 22-Dec-2020].
[2] “Scope of Variables,” *Scope of Variables · The Julia Language*. [Online]. Available: https://docs.julialang.org/en/v1/manual/variables-and-scoping/. [Accessed: 22-Dec-2020].
[3] “Functions,” *Functions · The Julia Language*. [Online]. Available: https://docs.julialang.org/en/v1/manual/functions/. [Accessed: 22-Dec-2020].
[4] S. O. Community, “Closures,” *Julia Language Tutorial*, 18-Sep-2016. [Online]. Available: https://julia-lang.programmingpedia.net/en/tutorial/5724/closures. [Accessed: 22-Dec-2020].