MUIC: Functional and Parallel Programming Midterm Quiz

Date: Tuesday, June 15th, 2021
Instructor: Rachata Ausavarungnirun

Problem 1 (30 Points):	
Problem 2 (25 Points):	
Problem 3 (35 Points):	
Problem 4 (20 Points):	
Total (110 points):	

Instructions:

1. This exam is designed to take 3 hours, but you have 30 hours on it.
2. You receive full credit if you score 100% points. Any points above 100% will be worth 0.5 (making the maximum score of 105)
3. Submit your work as a zip file on Canvas.
4. For the coding part, I expect everyone to **thoroughly test your code**. Hence, you must **submit the test cases for all the coding questions** along with the code and explain what each test case is used for. **Test cases, along with the explanation of how you test your code, will factor into 10% of all the coding questions.**
5. If not specified, input and output types are a part of the question. Please use appropriate input and output types that make sense for the purpose of the question.
6. Please clearly comment your code, especially if your code do not work perfectly,
7. Clearly indicate your final answer for each conceptual problem.
8. Our typical restriction for Scala packages are similar to Assignments 1 and 2.
9. **DO NOT CHEAT.** If we catch you cheating in any shape or form, you will be penalized based on **our plagiarism policy**.

Tips:

- **Read everything.** Read all the questions on all pages first and formulate a plan.
- **Be cognizant of time.** The submission site will close at 6PM on Wednesday.
- **Show work when needed.** You will receive partial credit at the instructors' discretion.

Initials: Manit Gandhi 6280004

1. Potpourri [30 points]

For this questions, please put your answers to parts a-d here and submit the pdf.

(a) **Folding** [5 points]

Use fold to sum all element in a list.

```
val lst = List(7,8,9,10)
val res = lst.fold(0)((x,y) => x+y)
println(res)
```

(b) **Benefit of CPS** [5 points]

What is the key benefit of CPS?

The key benefits of CPS are:-
1. It is simple
2. It eliminates tail recursion
3. It eliminates nested expressions
4. It reduces the complexity of high level program structures

(c) **Tail Recursion** [6 points]

Please write a factorial function using tail recursion.

```
def factorial(num:Int):Int = {
     def tailFact(num:Int, acc: Int):Int =
              if (num==0) acc else tailFact(num-1, acc*num)
     tailFac(num, 1)
  }
```

Initials: Manit Gandhi 6280004

(d) **CBV vs. CBN** **[8 points]**

Consider the following code.

```
def leftCBV(x:Int, y:Int) = x
def leftCBN(x:=> Int, y: =>Int) = x
def loop:int = loop

leftCBV(1+1,loop)        // Call 1
leftCBV(loop, 1+1)       // Call 2
leftCBN(1+1,loop)        // Call 3
leftCBN(loop, 1+1)       // Call 4
```

What happen to each of the four calls at the end? Please explain why each of them behave the way you observed using what we learned from the class.

For call 1, scala will try to evaluate the values that have been input. However, it will be like an indefinite iteration as the value loop will never be evaluated.

For call 2, scala will try to evaluate the values that have been input. However, it will be like an indefinite iteration as the value loop will never be evaluated.

For call 3, scala will substitute the values from the definition. leftCBN(1+1,loop) will return 2. When substitution is applied, loop is never used in the function.

Similarly call 4 will return loop

(e) **Types** **[6 points]**

From your Dessert in-class exercise, implement a function `blend(d: Dessert): String` that print `Blended [Dessert Name]` that is applicable for all types of Dessert (i.e., Pie, Cake and Smoothie).

Submit this code as `blended.scala`

Initials: Manit Gandhi 6280004

2. Product of Primes [25 points]

In this question, we will examine one of a very common technique used to share a private key.

Multiple two large prime.

(a) (5 points) Write a function `isPrime(n: Int): Boolean` that return true if `n` is a prime number and false otherwise. **Hint:** Feels free to use a helper function with more inputs.

(b) (5 points) Then, write a function `findPrimes(x:Int, y:Int): List[Int]` that takes in `x` and `y` and find all prime numbers between `x` and `y` (note that `x` and `y` is included in this range). **Hint:** Again, feels free to use a helper function with more inputs.

(c) (5 points) Then, write a function `profOfPrime():Int` that perform a multiplication on two random huge prime numbers (For this question, the function return a multiplication of random two primes between 10000000 and 20000000). Please use the functions you wrote in parts a and b for this.

(d) (10 points) You will find that the function can be slow because you ended up having to find all the primes between 10000000 to 20000000 before you can pick the two random primes. In this part, please use the concept of **Future** and **Await** to (hopefully) fix this problem. Please explain your solution as there are multiple good ways to do this. Be creative!

Please save the file with the name `prodOfPrimes.scala`

Initials: Manit Gandhi 6280004

3. Sorted List [35 points]

In this question, you are going to implement a more generic sorted list in Scala.

Below is the definition of our sorted list:

- A sorted list can either be `Empty` or contain multiple instances of elements that has total ordering property.
- For the purpose of implementing this list, you are not allowed to use the Scala's build-in List.
- For the purpose of getting things done, if you want to stick with a SortedList of Integer, feels free to do so but be aware that you will not be able to get a full credit for this question.

(a) (5 points) First, create a class called `SortedList` that contains all the necessary components that you need.

(b) (5 points) Implement an `insertListCPS(x:  A, L: SortedList):SortedList` function that **takes an input value and a SortedList**, and insert this value as a new element of the list while ensuring you preserve the SortedList's property. **This function should return a resulting SortedList after you insert the new element.**

(c) (5 points) Implement an `search(x:A, L:SortedList):Boolean` function that performs a search **for `x` on the input List** `L` and return true if `x` is in `L` and false otherwise.

(d) (5 points) Implement an `filterCPS(x:A, L:SortedList):SortedList` function that **takes an input `x` and return a List that contains only elements that is lower than `x`**. You must do this using the CPS style.

(e) (5 points) Implement a function `cleanUpListCPS(L: SortedList):SortedList` that takes an input SortedList type that does not always have the SortedList property. This function will drop any elements that break the sortedlist, or sort the entire list, and return the resulting list that is now a sorted list. For example, if your input is [1, 2, 4, 3, 5, 6, 7], your returned List can be [1, 2, 3, 5, 6, 7] or [1, 2, 4, 5, 6, 7], or [1, 2, 3, 4, 5, 6, 7].

(f) (5 points) In this part, make sure that your list works for any data types that have total ordering. If you do not want to attempt this question, then, make sure your other questions works with Integer.

(g) (5 points) Once everything is all done, please write your own test cases that thoroughly test your code **so that it also works in corner cases**. Your test should shows, as best as possible, that your implementation works as intended.

Please save the file with the name `sortedlist.scala`

Initials: Manit Gandhi 6280004

4. Scope [20 points]

In this question, we will use a pseudocode below that does not exactly compile in Scala to test your understanding of the lexical scope and dynamic scope. Consider the following pseudocode. In this question, let us assume emphval is allowed to remap the variable name (i.e., x, y, z) to the newly defined expression and the value used inside the expression follow the lexical or dynamic scope rules from our lecture.

If the expression does not evaluate to a value, please state why. If the expression break the type rule, please explain why.

Hint: Your answers might be a function (functions are values!).

```
val x = 10                                         // Line 1
val y = 20                                         // Line 2
def f(x:Int) = x + y                               // Line 3
def g(y:Int) = (z:Int) => f(z) + f(x) - y - z     // Line 4
val x = f(x) + x                                   // Line 5
val x = g(x) + x                                   // Line 6
val z = x(y)                                       // Line 7
```

(a) What is the difference between Lexical and Dynamic scope? (4 points)

When the location of a function's definition determines which variables we have access to, it is referred to as Lexical scope.

However, a dynamic scope uses the location of the function's invocation to determine the availability of variables.

(b) Assuming we use **Lexical Scope**, what is the value of x at line number 7? Briefly explain why. (4 points)

Value of x will be 100 as evaluated in line 6

Initials: Manit Gandhi 6280004

(c) Assuming we use **Lexical Scope**, what is the value of y at line number 7? Briefly explain why. (4 points)

The value of y will be 20 because y has not been updated since its evaluation on line 2.

(d) Assuming we use **Lexical Scope**, what is the value of z at line number 7? Briefly explain why. (4 points)

z will be 40 as it was evaluated in line 6 by g(x)

(e) Assuming we use ***Dynamic Scope***, what is the value of z at line number 7? Briefly explain why. (4 points)

Since x(y) is not a function, z will not be evaluated