Basics of Scala

Vicente De Leon

UID: 2001014594

Assignment 3

1. Question 1

Resources (Please check Scala Worksheet Resources for more detail):

* Reading input: <https://docs.scala-lang.org/overviews/scala-book/command-line-io.html>
* Taking input from user: <https://stackoverflow.com/questions/5055349/how-to-take-input-from-a-user-in-scala>
* Print vs println: <https://www.tutorialspoint.com/difference-between-print-and-println-in-java>
* Initializing variable: <https://stackoverflow.com/questions/2754301/initialise-a-var-in-scala>
* While loop: <https://stackoverflow.com/questions/46199788/java-while-loop-for-entering-username-and-password-3-times-issue-with-break-s>
* If/Else statement: <https://www.tutorialspoint.com/scala/scala_if_else.htm>

Observations:

* scala.io.StdIn.readLine is going to be used in order to read command\_line input (easiest way to do it).
* Username and password are provided within Question1 Scala program.
* Since we are working with console input, “print()” is going to be also used.
* Basic Scala program that asks the user to enter a username (vdeleonw) and a password (ScalaBasics).
* Basic while loop will be implemented to let the user enter the password until it matches the username (similar to Python).

Please, see following page for images.

Question 1 images (inputs and outputs)

A screenshot of a computer program

Description automatically generated with medium confidence

A black screen with white text

Description automatically generated with low confidence

A screen shot of a computer

Description automatically generated with medium confidence

1. Question 2

Resources (Please check Scala Worksheet Resources for more detail):

* Following question 1 resources.
* Scala Decision making: <https://www.geeksforgeeks.org/scala-decision-making-if-if-else-nested-if-else-if-else-if/>
* For loop: <https://www.geeksforgeeks.org/for-loop-in-scala/>
* String interpolation: <https://docs.scala-lang.org/overviews/core/string-interpolation.html>
* Lower case vs caps lock: <https://www.geeksforgeeks.org/scala-string-equalsignorecase-method-with-example/>

Observations:

* We are using same resources as question plus a couple of new resources that help building the for loop, if, else, and else if statements. It is important to state that this is the expansion of question 1.
* In this question we are including new steps to ensure the privacy of our program. These steps include a security question “What class did you take last semester?” and the answer “Applied Machine Learning” both being private values in Scala.
* Again we use readLine() and print for user input console purposes and we expand the logic behind the program to include the following:
* A for loop to let the user enter password up to 3 times until it matches the username.
* Resetting the password if the three attempts fail.
* Choosing between Yes or No, even if the user chooses to write in lower case. If he or she chooses yes or Yes, a security question will pop up asking a personal question.
* There’s also a message integrated in case the answer for the security question fails.

Please, see following page for images.

Question 2 images (inputs and outputs)

A screen shot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated with medium confidence



1. Question 3

Resources (Please check Scala Worksheet Resources for more detail):

* Tutorial code Python version: <https://github.com/doocs/leetcode/blob/main/solution/0000-0099/0029.Divide%20Two%20Integers/README_EN.md>
* Operators: <https://www.tutorialspoint.com/scala/scala_operators.htm>
* Math.abs(): <https://www.alphacodingskills.com/scala/notes/scala-math-abs.php>
* toDouble: <https://www.includehelp.com/scala/convert-int-to-double.aspx>

Observations:

* This code is the Scala version of the Python code within the Tutorial code Python version resource. The reason I decided to take this count into account is due to the simple approach it takes to generate the desire results. Also, is worth mentioning that I feel comfortable using Python as I use it daily. The following images explain the Python version:



A screenshot of a computer

Description automatically generated with medium confidence

* We are going to try and perform a division between “a” and “b”. MAX and MIN are just the maximum and minimum values of an integer. The sign has a similar but different syntax than the python version due to the if else statements in Scala.
* The Math.abs() is used to calculate the absolute values (positive values). As stated in the code below, many operators like the binary left shift operator, the is false (&&), is not true, is true, and assignment operators follow the same pattern as the python version. The loop is being used to perform the division, subtracting the divisor (y) from the dividend (x). The toDouble was added to convert the integer value into double value (4.6666666667).

Please, see following page for images.

Question 3 images (inputs and outputs)

A screenshot of a computer program

Description automatically generated with medium confidence

A black background with white text

Description automatically generated with low confidence

1. Question 4

Resources:

* Code tutorial: <https://www.geeksforgeeks.org/all-unique-combinations-whose-sum-equals-to-k/>
* Scala Lists: <https://www.geeksforgeeks.org/scala-lists/>
* ListBuffer: <https://www.scala-lang.org/api/2.13.6/scala/collection/mutable/ListBuffer.html>
* ListBuffer: <https://www.geeksforgeeks.org/scala-listbuffer/>
* mkString(): <https://www.geeksforgeeks.org/scala-list-mkstring-method-with-a-separator-with-example/>
* operators: https://www.tutorialspoint.com/scala/scala\_operators.htm

Observations:

* Just like question 3, this code follows a python version to find all unique combinations of given elements such that their sum is K.
* The Scala code generates unique combinations of numbers from a given list (assignment example) that its addition is equal to the desired target. In this case the list [10,1,2,7,6,1,5] and the target value is 8. So, if the sum equals to target 8, results in “local” list will be print out. We are going to be using the Scala ListBuffer() is going to be used to store and modify data (mutable data).
* We are going to be checking for logical condition and duplicates using the following code: if (newSum <= K && !(i > l && A(i) == A(i - 1))).
* We proceed to recursive call the first function, after adding elements, and then remove last element of local list.
* We use main method() to call the uniqueCombination() function to both A list and target vale K.

Please, see following page for images.

Question 4 images (inputs and outputs)

A picture containing text, screenshot

Description automatically generated

A black screen with white text

Description automatically generated with low confidence

1. Question 5

Resources:

* Arrays: <https://www.geeksforgeeks.org/scala-arrays/>
* Distinct: <https://www.geeksforgeeks.org/scala-list-distinct-method-with-example/>
* mkString(): <https://www.geeksforgeeks.org/scala-list-mkstring-method-with-a-separator-with-example/>

Observations:

* This is basic and self-explanatory. We have a list, and we want to avoid duplicates.
* Question 5 images (inputs and outputs):

A picture containing text, screenshot, font

Description automatically generated

A black background with white text

Description automatically generated with low confidence

1. Question 5

Resources (Please check Scala Worksheet Resources for more detail):

* List: <https://www.geeksforgeeks.org/scala-lists/>
* ::: method: <https://www.includehelp.com/scala/merge-lists-in-scala.aspx>
* Descending order: <https://blog.knoldus.com/sorting-in-scala-using-sortedsortby-and-sortwith-function/>

Observations:

* Last question is also very basic, it’s about merging two lists.
* Question 6 images (inputs and outputs):

A picture containing text, screenshot, font

Description automatically generated

A black background with white text

Description automatically generated with low confidence

Codes:

//General: https://www.geeksforgeeks.org/scala-programming-language/

import scala.io.StdIn.readLine // easiest way to read command\_line input

import scala.collection.mutable.ListBuffer // creating mutable data

// Question 1

object HW3Question1 {

val username = "vdeleonw"

val password = "ScalaBasics"

def main(args: Array[String]): Unit = {

print("Please enter your username: ")

val existingUsername = readLine() // read from the console

if (existingUsername == username) {

var existingPassword = "" // initialize empty string

while (existingPassword != password) {

print("Please enter your password: ")

existingPassword = readLine() // read from the console

if (existingPassword == password) {

println("Welcome vdeleonw!")

} else {

println("Incorrect password. Please try again.")

}

}

} else {

println("Username not valid or doesn't exists. Please run the program again.")

}

}

}

// Question 2

object HW3Question2 {

val username = "vdeleonw"

var password = "ScalaBasics"

private val question = "What class did you take last semester?"

private val answer = "Applied Machine Learning"

def main(args: Array[String]): Unit = {

print("Please enter your username: ")

val existingUsername = readLine() // reading user input console

if (existingUsername == username) {

var attempts = 0 //initializing local variable

for (attempts <- 1 to 3) { // for loop, range from 1-3

print("Please enter your password: ")

val existingPassword = readLine() // reading user input console

if (existingPassword == password) {

println("Welcome vdeleonw!")

return

} else if (attempts < 3) { // deciding amon options geek for geeks

println("Incorrect password. Please try again.")

}

}

print("Would you like to reset your password? Please choose Yes or No: ")

if (readLine().equalsIgnoreCase("yes")) { // equalsIgnoreCase -> either lower case or cap locks

print(s"$question: ") // string interpolation

if (readLine() == answer) {

print("Please enter your new password: ")

password = readLine() // reading user input console

println("Done. Your password has been reset.")

} else {

println("The answers you provided are not correct. Your account is blocked.")

}

}

} else {

println("Username not valid or doesn't exist. Please run the program again.")

}

}

}

// Question 3

object HW3Question3 {

private def divide(a: Int, b: Int): Int = {

val INT\_MAX = (1 << 31) - 1 // << -> binary left shift operator

val INT\_MIN = -(1 << 31) // // << -> binary left shift operator

val sign = if (a \* b < 0) -1 else 1

var new\_a = Math.abs(a) // specify a number whose absolute value need to be determined.

val new\_b = Math.abs(b) // specify a number whose absolute value need to be determined.

var tot = 0

while (new\_a >= new\_b) { // new\_a >= new\_b is not true

var cnt = 0

do {

cnt += 1 // assignment just like python

} while (new\_a >= (new\_b << cnt))

cnt -= 1 // assignment just like python

tot += 1 << cnt

new\_a -= new\_b << cnt

}

val ans = sign \* tot

if (INT\_MIN <= ans && ans <= INT\_MAX) ans else INT\_MAX // (INT\_MIN <= ans && ans <= INT\_MAX) is false

}

def main(args: Array[String]): Unit = {

val x = 14 // dividend

val y = 3 // divisor

val division = divide(x, y)

val result = x.toDouble / y.toDouble

println(f"Explanation: $x/$y = $result%.10f. which is truncated to $division.") // string interpolation

}

}

// Question 4

object HW3Question4 {

def uniqueCombination(l: Int, sum: Int, K: Int, local: ListBuffer[Int], A: List[Int]): Unit = {

if (sum == K) { // if sum equals k print results within local

println(local.mkString(",")) // aesthetic purposes mkString separation with ,

return

}

for (i <- l until A.length) { // iterate over elements

val newSum = sum + A(i) // newSum contains the values from "sum"

if (newSum <= K && !(i > l && A(i) == A(i - 1))) { // checks if newSum is less than or equal to target k and it check for duplicates

local += A(i) // local list containing combinations

uniqueCombination(i + 1, newSum, K, local, A) // (recursive) implement above function after adding elements

local.remove(local.length - 1) // remove last element of local list

}

}

}

def main(args: Array[String]): Unit = {

val A = List(10, 1, 2, 7, 6, 1, 5) // assignment list example

val K = 8 // target example

uniqueCombination(0, 0, K, ListBuffer(), A.sorted) // mutable data with ListBuffer and sorting list A for aesthetic purposes

}

}

// Question 5

object HW3Question5 {

def main(args: Array[String]): Unit = {

val nums = Array(0,0,1,1,1,2,2,3,3,4)

val noDuplicates = nums.distinct // simpler approach to avoid duplicates within list "num"

println(noDuplicates.mkString(", ")) // mkString for aesthetic results

}

}

// Question 6

object HW3Question6 {

def main(args: Array[String]): Unit = {

val list\_1 = List(14, 12, 25, 0, 1)

val list\_2 = List(123, 2, 44, 3)

val list\_3 = list\_1 ::: list\_2 // Merging 2 lists using the ::: method

val FinalList = list\_3.sortWith(\_ > \_) // SortWith() function

println(FinalList)

}

}